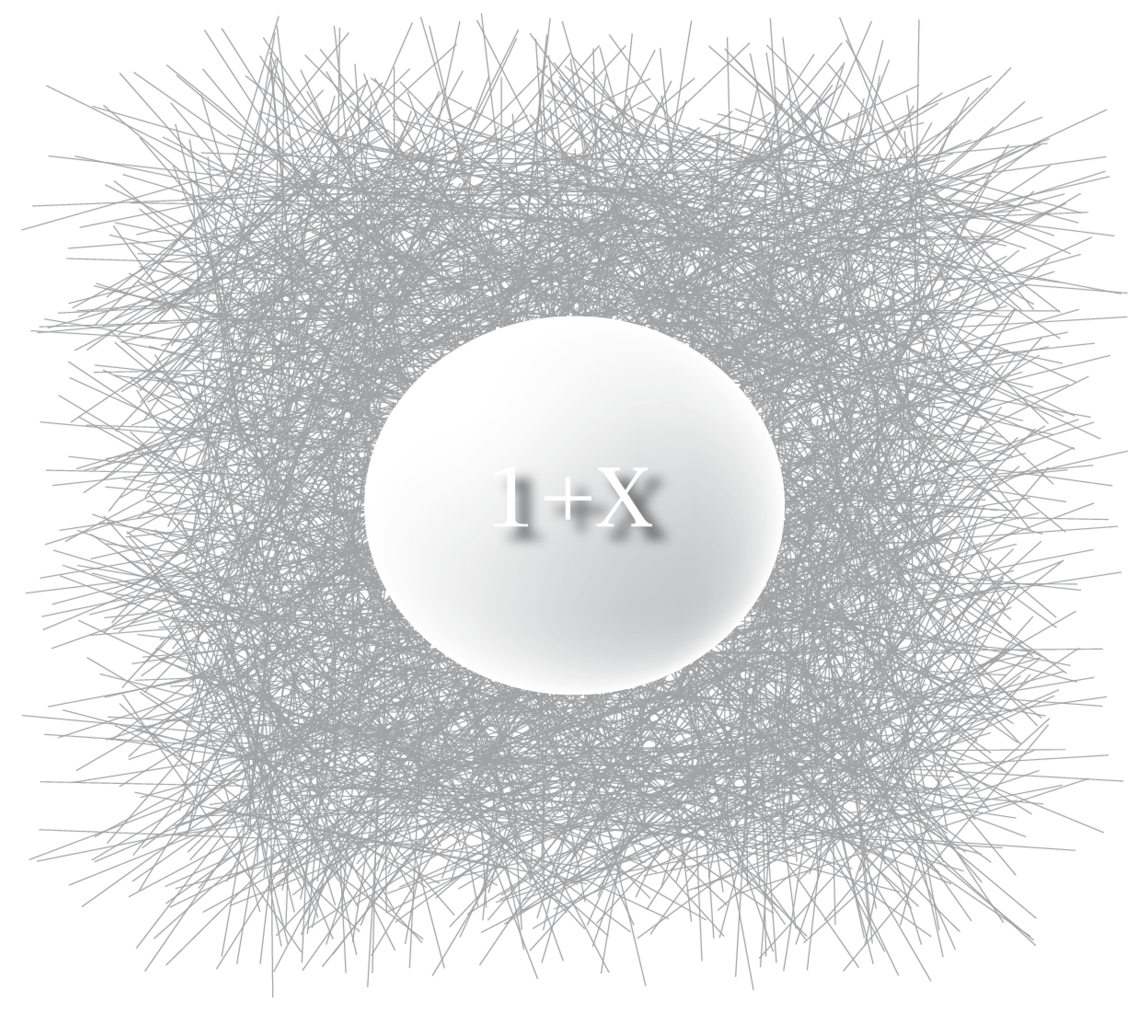
大学计算机基础教育规划教材

Python程序设计习题解析

周元哲　编著



清华大学出版社

大学计算机基础教育规划教材

Python 程序设计习题解析

周元哲　编著

清华大学出版社
北　京

内容简介

本书与《Python 程序设计基础》(周元哲编著)配套,针对该教材各章节中的内容,介绍每章的知识点和重点、难点知识,给出了每章的课后习题答案,并精心设计和安排了相应的习题解析。另外,本书新增 4 章内容:网络爬虫、软件测试框架、Web 开发框架、游戏开发。

本书可作为高等院校各专业程序设计和软件竞赛培训的辅导教材,也可作为程序员、编程爱好者的自学辅助用书和各类培训班的教学辅助用书。

图书在版编目(CIP)数据

Python 程序设计习题解析/周元哲编著. —北京:清华大学出版社,2017
(大学计算机基础教育规划教材)
ISBN 978-7-302-46649-9

Ⅰ. ①P…　Ⅱ. ①周…　Ⅲ. ①软件工具—程序设计—高等学校—题解　Ⅳ. ①TP311.561-44

中国版本图书馆 CIP 数据核字(2017)第 032050 号

责任编辑: 张　民　战晓雷
封面设计: 常雪影
责任校对: 胡伟民
责任印制: 刘海龙

出版发行: 清华大学出版社
网　　址: http://www.tup.com.cn, http://www.wqbook.com
地　　址: 北京清华大学学研大厦 A 座　**邮　编:** 100084
社 总 机: 010-62770175　**邮　购:** 010-62786544
投稿与读者服务: 010-62776969,c-service@tup.tsinghua.edu.cn
质 量 反 馈: 010-62772015,zhiliang@tup.tsinghua.edu.cn
课 件 下 载: http://www.tup.com.cn,010-62795954
印 装 者: 保定市中画美凯印刷有限公司
经　　销: 全国新华书店
开　　本: 185mm×260mm　**印　张:** 10.5　**字　数:** 260 千字
版　　次: 2017 年 5 月第 1 版　**印　次:** 2017 年 5 月第 1 次印刷
印　　数: 1~2000
定　　价: 29.00 元

产品编号:073840-01

前 言

Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言,被列入 LAMP(Linux、Apache、MySQL 以及 Pthon/Perl/PHP),Python 语言将代码书写缩进作为语法规定,使得初学者养成良好的编码习惯。Python 具有序列、列表、元组和字典等数据结构,便于实现各种算法。作为开源语言,Python 外挂各种库,在大数据、科学计算等方面,功能堪比 Matlab。因此,作为面向对象的程序设计语言,Python 具有简单、现代、类型安全、性能优良,效率高等特点,是计算机等信息学科学生学习程序设计语言的首选之一。

本书与《Python 程序设计基础》(周元哲编著)相配套,针对《Python 程序设计基础》各章(Python 编程概述、数据类型和表达式、顺序与选择结构、循环结构、序列与字典、数据结构与算法、函数与模块、面向对象程序设计基础、文件、用户界面设计、绘图、数据库应用、网络编程和异常处理)中的内容,介绍每章的知识重点,给出每章的课后习题答案,并精心设计和安排了相应的习题与解答。另外,本书新增 4 章内容,包括网络爬虫、软件测试框架、Web 开发框架、游戏开发。

计算机学院李晓戈、邓万宇等阅读了部分手稿。西北工业大学软件与微电子学院郑炜对本教材的写作大纲、写作风格等提出了很多宝贵的意见。衷心感谢上述各位的支持和帮助。本书在写作过程中参阅了大量中英文专著、教材、论文、报告及网上的资料,限于篇幅,未能一一列出。在此,向有关作者一并表示敬意和衷心的感谢。

本书内容精练,文字简洁,结构合理,实训题目经典实用,综合性强,明确定位于面向初、中级读者,由入门起步,侧重提高。特别适合作为高等院校相关专业 Python 程序设计的教材或教学参考书,也可以供从事计算机应用开发的各类技术人员参考。

本书的 Python 版本为 2.7.3,所有程序都在 PyCharm 编辑器中进行了调试和运行。由于作者水平有限,时间紧迫,本书难免有疏漏之处,恳请广大读者批评指正。

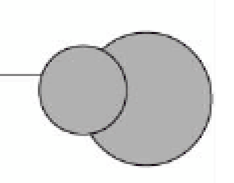
作者的电子信箱是 zhouyuanzhe@ 163.com。

作　者

2016 年 11 月

目　录

第1章 Python编程概述



1.1 本章要求

- 了解计算机基础知识。
- 了解 Python 语言及其特点。
- 了解 Python 语言版本和开发环境。
- 掌握 Python 解释器的下载和安装。
- 掌握 Python 各类编辑器，特别是 Pycharm。
- 了解 Python 与其他语言的关系。

1.2 本章知识重点

1.2.1 Python 的特点

Python 是一种简单易学、功能强大的编程语言，它有高效率的高层数据结构，能够简单而有效地实现面向对象编程。Python 具有如下一些特点。

1. 简单易学

Python 语法简捷而清晰，易于快速上手学习，在学习过程中便于专注程序本身的逻辑和算法，探究程序执行的过程。

2. 免费开源

Python 是开源软件，可以自由地阅读它的源代码。

3. 解释型语言

Python 作为解释型语言，其源代码通过解释器转换成字节码的中间形式，由虚拟机在不同计算机上运行。

4. 面向对象

Python 是面向对象的语言，函数、模块、字符串都是对象，并且完全支持继承、重载、

派生、多重继承。

5. 丰富的库

Python 称为胶水语言，能够轻松地与其他语言(特别是 C 或 C ++)结合在一起，具有丰富的 API 和标准库。

1.2.2 Python 解释器

下面介绍 Python 解释器在 Linux 和 Windows 下的安装步骤。

1. 在 Ubuntu 下安装 Python

Ubuntu(乌班图)是一个以桌面应用为主的 Linux 操作系统，内置 Python，如图 1.1 所示。

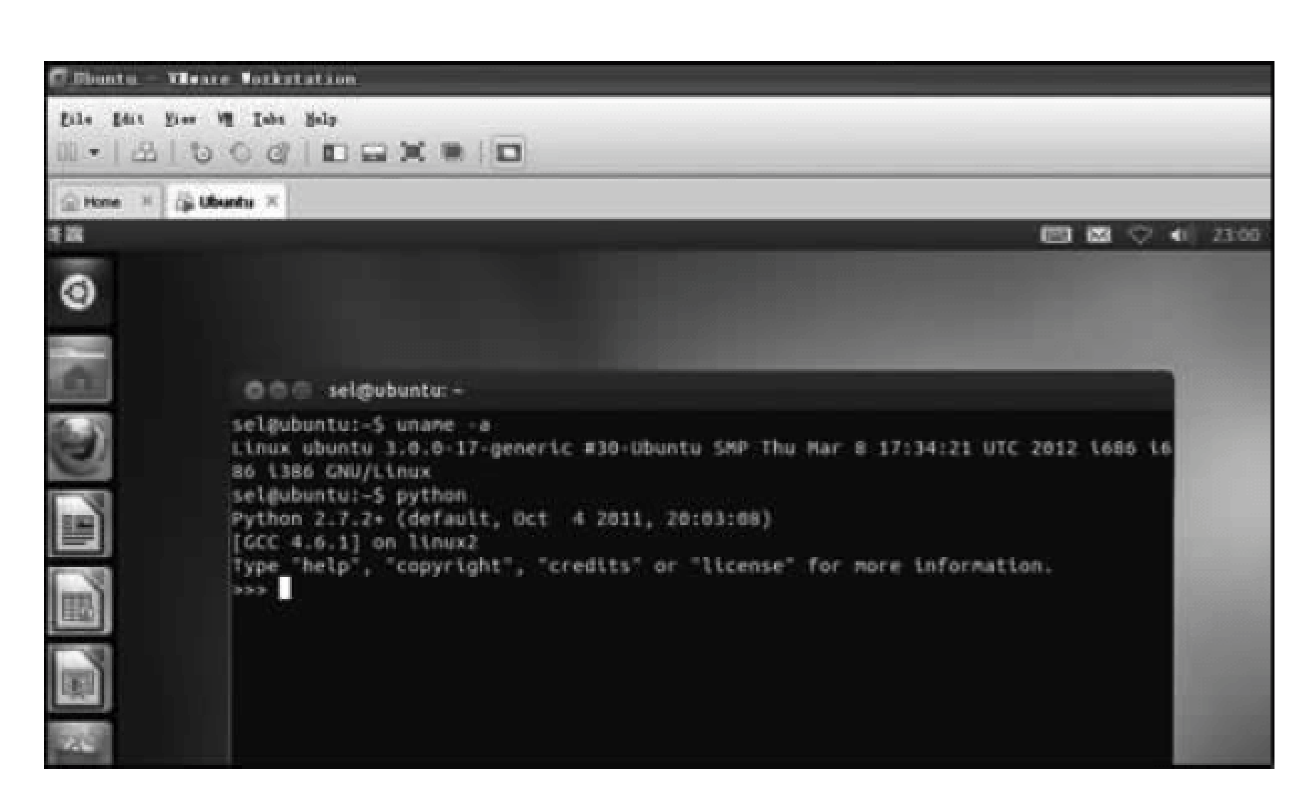


图 1.1 Ubuntu 下内置 Python

2. 在 Windows 下安装 Python

在 Windows 下安装 Python，一般具有如下步骤。

步骤一：下载 Python 2.7.3 安装包进行安装。

在浏览器中输入 http://www.python.org，在下载页 https://www.python.org/download/releases/2.7.3/中找到 Windows x86 MSI Installer (2.7.3) (sig)进行下载，如图 1.2 所示。

步骤二：在 Windows 环境变量中添加 Python。

将 Python 的安装目录添加到 Windows 下的 path 变量中，如图 1.3 所示。

步骤三：测试 Python 安装是否成功。

在 Windows 下使用 cmd 打开命令行窗口，输入 python 命令测试 Python 安装是否成功，图 1.4 表示安装成功。

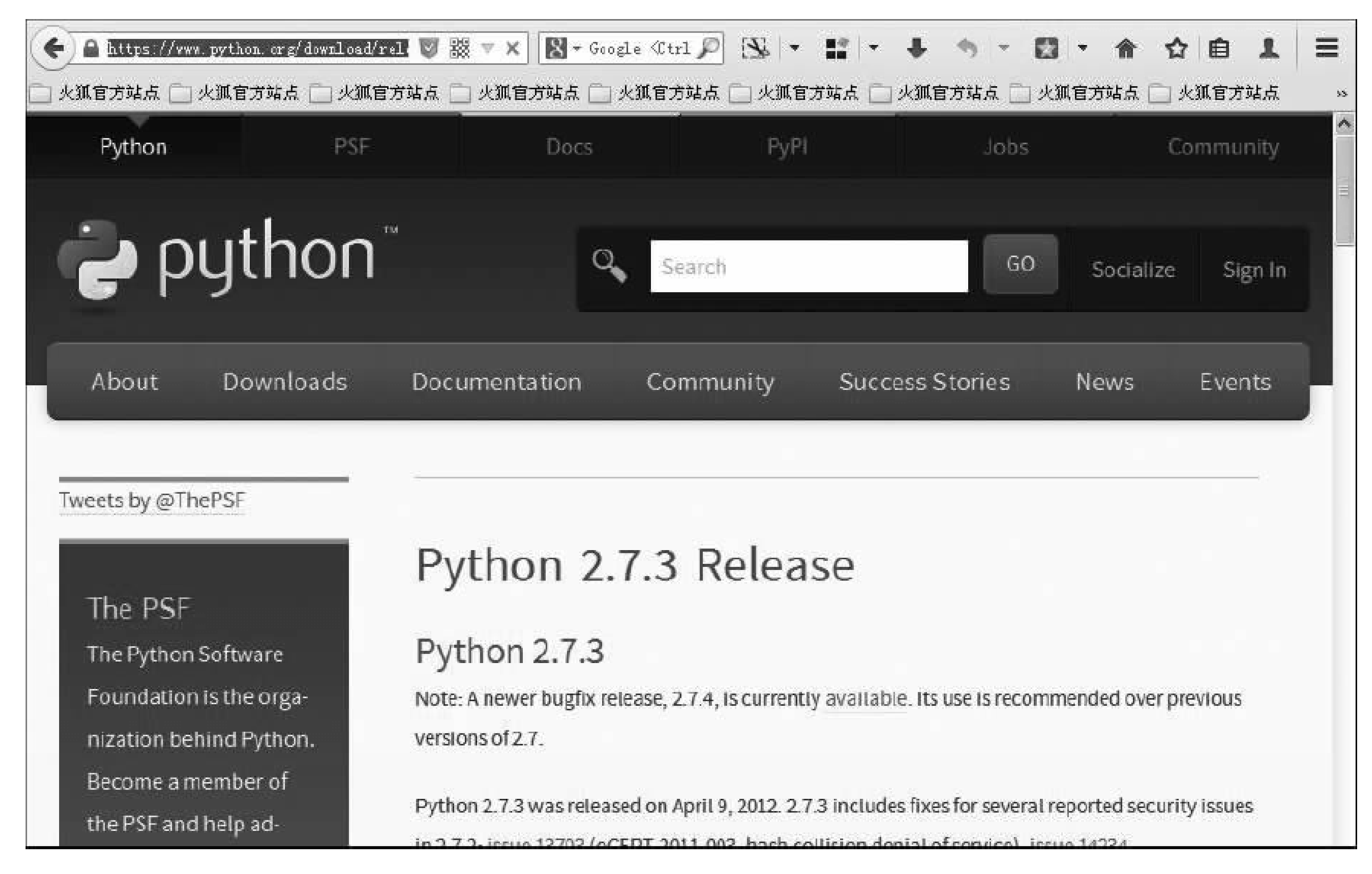


图 1.2 下载 Python 2.7.3



图 1.3 设置环境变量

```
C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'hello'
hello
>>>
```

图 1.4　测试 Python 安装是否成功

1.2.3　Python 编辑器

Python 编辑器众多，有 Python 自带的 IDE 编辑器、Notepad＋＋、UliPad 以及 Vim 和 emacs 等。Linux 下的 Eclipse with PyDev 和 Windows 下 Python 编辑器 PyCharm 功能较为强大，本书重点介绍 PyCharm。PyCharm 具有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，PyCharm 提供了一些高级功能，支持 Django 框架下的专业 Web 开发。下载 PyCharm 双击安装，如图 1.5 所示。



图 1.5　安装 PyCharm 步骤 1

单击 Next 按钮，如图 1.6 所示。

安装结束，运行 PyCharm，如图 1.7 所示。

下一步，可以选择免费试用 30 天，如图 1.8 所示。

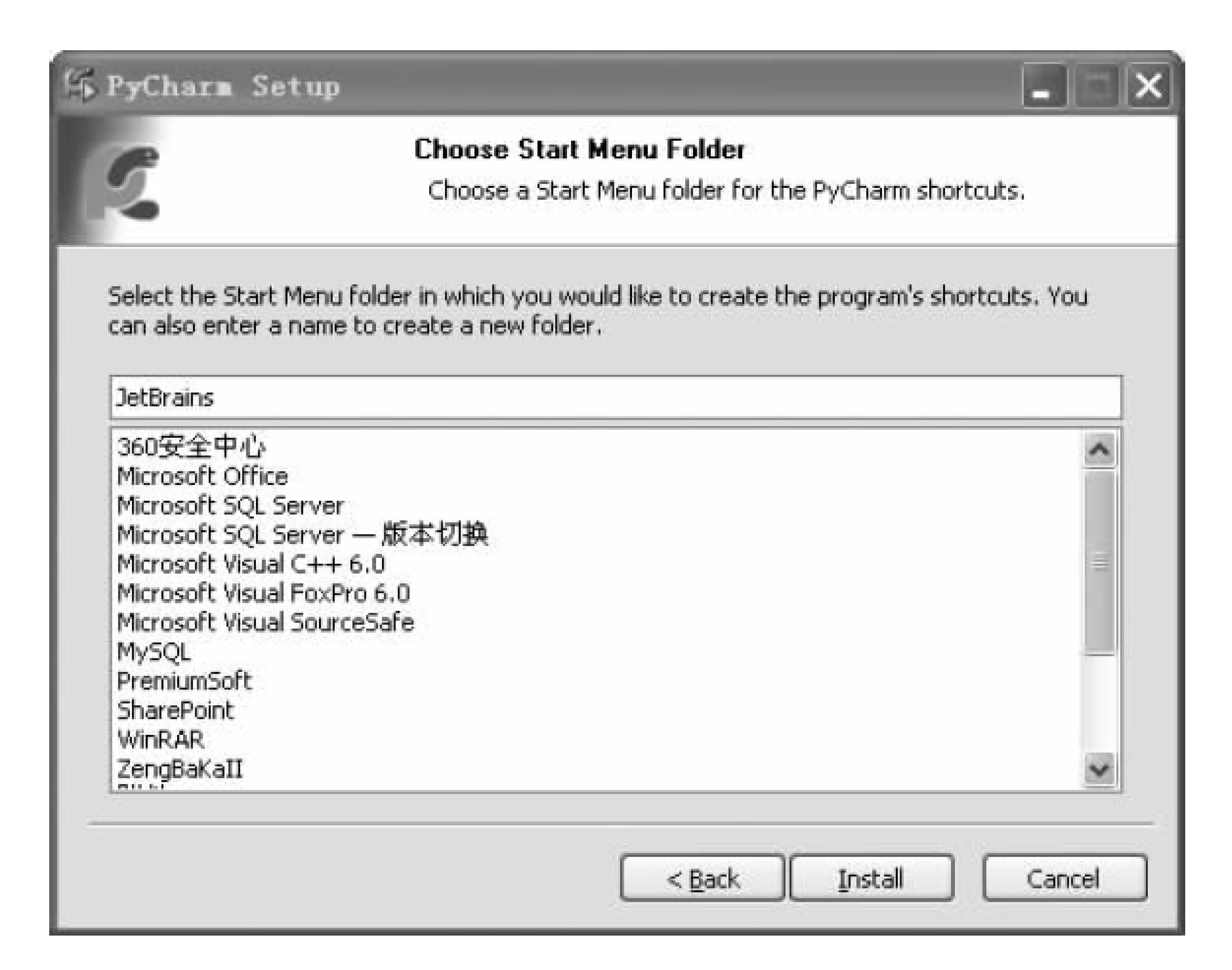


图 1.6　安装 PyCharm 步骤 2

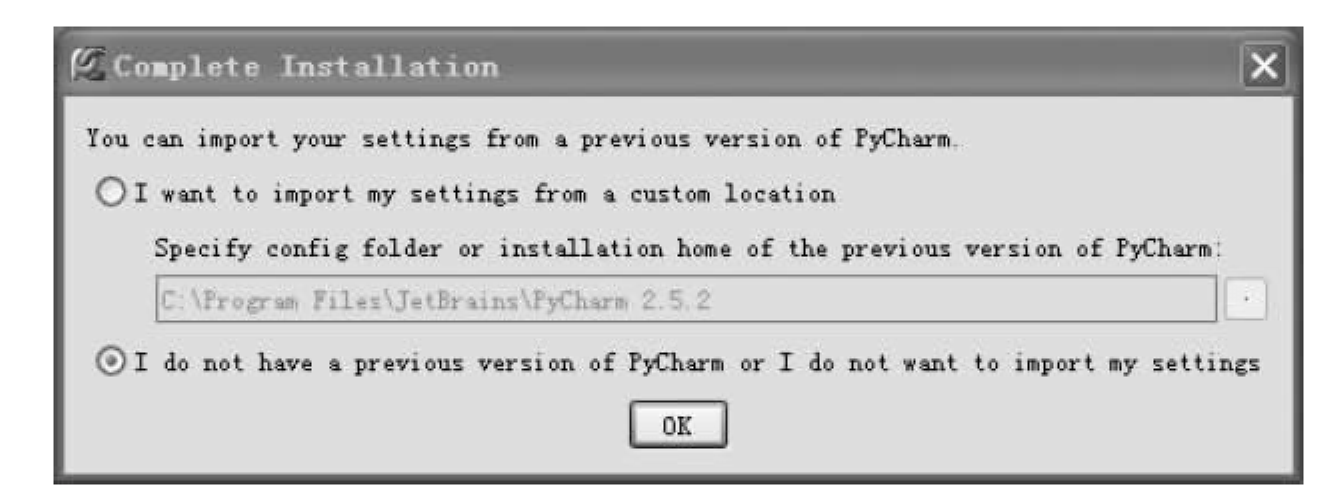


图 1.7　运行 PyCharm

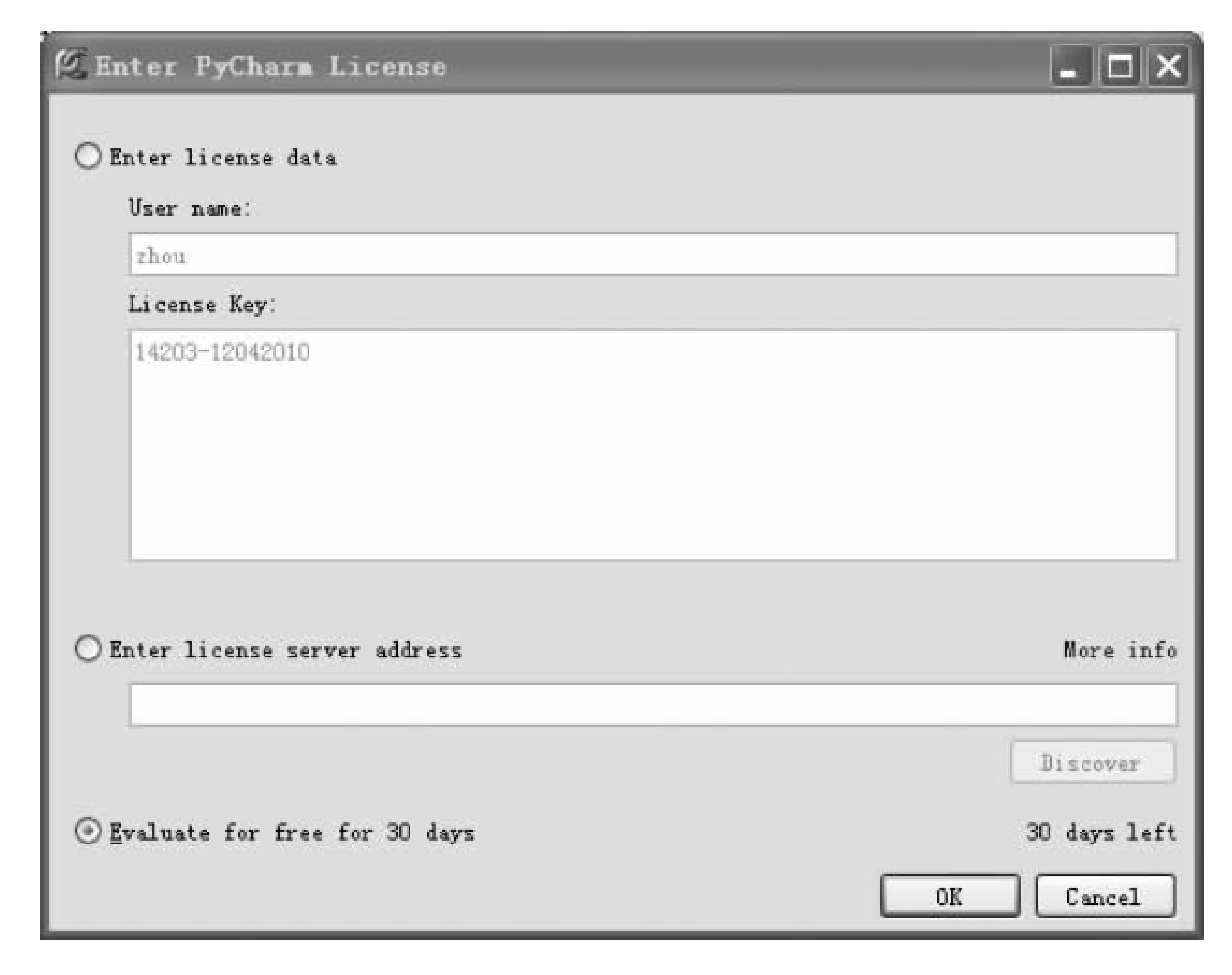


图 1.8　激活 PyCharm

单击 Create New Project，输入项目名、路径，选择 Python 解释器。如果没有 Python 解释器，对话框如图 1.9 所示。



图 1.9 选择 Python 解释器

选择 Python 解释器，本书用的是 python-2.7.3.msi，如图 1.10 所示。



图 1.10 完成 PyCharm 安装

启动 PyCharm，创建 Python 文件，如图 1.11 所示。

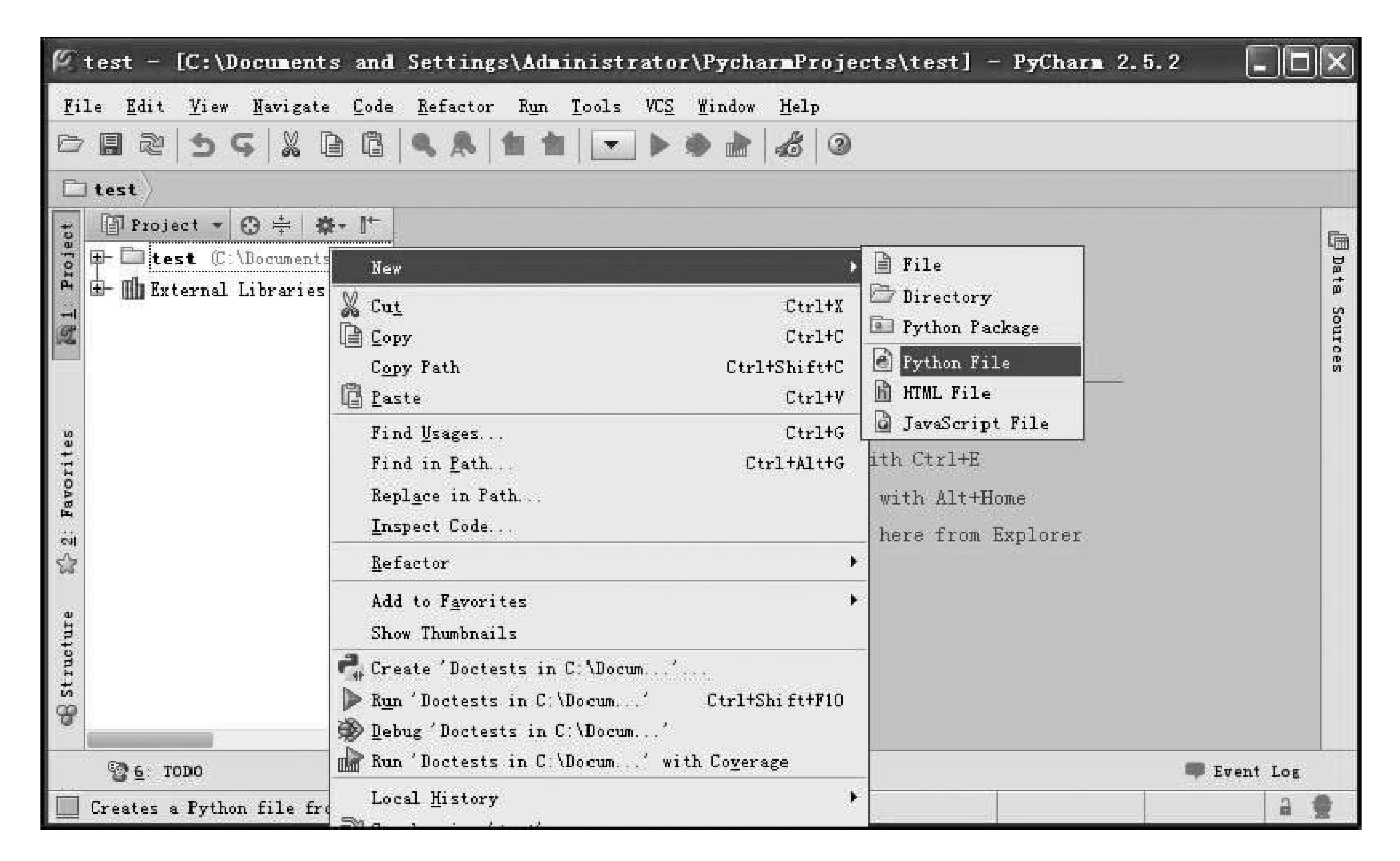


图 1.11　在 PyCharm 中创建 Python 文件

1.3　课后习题答案

1. 冯・诺依曼理论是什么?

【解答】 冯・诺依曼理论有以下两个要点:

(1) 计算机硬件设备由存储器、运算器、控制器、输入设备和输出设备 5 个部分组成。其中,运算器和控制器组成中央处理器单元(Center Process Unit,CPU)。中央处理单元用于执行指令,如算术操作、从别的设备写入或读出数据。存储器分为内存和外存。CPU 从内存中读取所需要的数据,进行处理。内存中存储的数据是临时的,当程序退出或者计算机关机时,数据将会丢失。如果需要永久地存储数据,需要用到外存,如硬盘、闪存等设备。键盘、鼠标等输入设备用于接收用户输入数据和指令,显示器通常作为输出设备。

(2) 存储程序思想。把计算过程描述为由许多命令按一定顺序组成的程序,然后把程序和数据一起输入计算机,计算机对已存入的程序和数据进行处理后输出结果。

2. 软件和程序是否一样?

【解答】 一般认为,软件包括以下一些内容:

(1) 运行时,能够提供所要求的功能和性能的指令或计算机程序集合。

(2) 程序能够令人满意地处理信息的数据结构。

(3) 有描述程序功能需求、程序如何操作和使用所要求的文档。

软件和程序是两个概念,初学者往往会混淆这两者。其实,这发生在软件历史的第一阶段(20 世纪 50 年代初期至 60 年代中期),由于软件的生产个体化,规模较小,功能单一,软件只有程序而无文档,形成了“软件等于程序”的错误观念。程序是为实现特定目标或解决特定问题而用计算机语言编写的命令序列的集合,通过使用与自然语言具有相似

的语法和语义的程序设计语言编写源代码，利用特定的工具将其翻译成 CPU 所能执行的指令，完成特定的目的。

3. 程序设计语言经过了哪些阶段？

【解答】 程序设计语言的发展经过了以下几个阶段。

(1) 第一代程序设计语言。

机器语言是用二进制代码表示的计算机能直接识别和执行的一种机器指令的集合，指令是由 0 和 1 组成的一串代码，通过线路变成电信号，让计算机执行各种不同的操作。机器语言具有直接执行的特点。编程人员需要熟记所用计算机的全部的由若干个 0 和 1 组成的指令代码及其含义，机器语言具有难读、难编、难记和易出错的缺点。

(2) 第二代程序设计语言。

为了克服机器语言的缺点，人们用与代码指令实际含义相近的英文缩写词、字母和数字等符号来取代指令代码(如用 ADD 表示运算符号“＋”的机器代码)，采用助记符号编写程序，于是就产生了汇编语言。汇编语言比用机器语言的二进制代码编程要方便些，在一定程度上简化了编程过程。

汇编语言又称为符号语言，不能直接被计算机识别，要使用一种程序将汇编语言翻译成机器语言。当汇编语言产生面向硬件的操作控制信息的指令时，使用起来依旧烦琐，程序无结构，通用性也差。但是，使用汇编语言编制系统软件和过程控制软件，其目标程序占用内存空间少，运行速度快，有着高级语言不可替代的用途。

(3) 第三代程序设计语言。

由于机器语言、汇编语言依赖于硬件体系，要求使用者必须对硬件结构及其工作原理都十分熟悉，因此人们又发明了与人类自然语言相接近且能为计算机所接受的规则明确、通用易学的计算机语言，其语法和结构具有类似文字的表现形式。1954 年，第一个面向科学计算的高级计算机语言——Fortran 语言被正式推广使用，Fortran 是 Formula Translation 的缩写，意为“公式翻译”，是数值计算领域所使用的主要语言。1972 年，作为程序语言的里程碑出现的 C 语言不但具有高级语言的特点，又具有汇编语言的特点，逐渐成为教学科研和软件开发的主要语言。

(4) 第四代程序设计语言。

面向对象程序设计语言、脚本语言、人工智能语言等通常被认为是第四代程序设计语言。SIMULA67 是第一个面向对象程序设计语言，特别是 1995 年 5 月由 Sun 公司推出的 Java 程序设计语言，可以撰写跨平台应用软件。第四代程序设计语言提供了功能强大的非过程化问题定义手段，用户只需告知系统做什么，而无须说明怎么做，因此可大大提高软件生产效率。Python 就属于第四代程序设计语言。

4. 简述 Python 的功能和特点。

【解答】 Python 是一种简单易学、功能强大的编程语言，它有高效率的高层数据结构，能简单而有效地实现面向对象编程。Python 具有简单易学、免费开源、解释执行、面向对象、丰富的库等特点。

5. 安装 Notepad＋＋，学习如何使用。

【解答】 略

6. 安装 UliPad，学习如何使用。

【解答】 略

7. 安装 Eclipse+PyDev，学习如何使用。

【解答】 略

8. Python 相比其他程序设计语言有什么特点？

【解答】 下面依次介绍 Python 与 C 和 Java 的关系。

Python 语言与 C 语言的区别如下：

(1) Python 是动态编译语言，变量的使用只需赋值；C 语言是静态编辑语言，变量的使用必须先定义，并显式说明其类型方法能使用。

(2) Python 有列表、元组、字典等数据结构，而 C 语言没有。

(3) Python 是弱类型语言，C 语言是强类型语言。

Python 语言与 Java 语言的关系如下：

(1) Python 和 Java 都支持面向对象编程。

(2) Python 比 Java 要简单，非常适合构造快速原型。

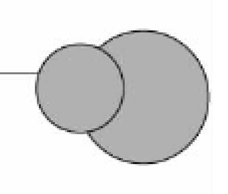
(3) Python 和 Java 都适合程序员协同开发大型项目。

(4) Python 和 Java 都语法简洁，表达能力强，而同样的工作，Python 只需要 Java 的 1/3 代码量。

总之，Python 具有如下特点：

- 比 TCL 强大，支持大规模编程，适于开发大型系统。
- 比 Perl 语法简洁，更具可读性，更易于维护，有助于减少 Bug。
- 比 Java 更简单，更易于使用。
- 比 C 和 C++更简单，更易于使用。Python 不用做很多底层工作，可以进行快速模型验证。
- 比 VB 更强大，也更具备跨平台特性。
- 比 Ruby 更成熟，语法更具可读性。

第2章 数据类型和表达式



2.1 本章要求

- 了解数据类型。
- 掌握标识符及其命名规则。
- 掌握变量和赋值。
- 掌握运算符。
- 掌握表达式。
- 掌握系统函数。
- 掌握 Python 字符。

2.2 本章知识重点

2.2.1 数据类型

Python 提供的基本数据类型主要有布尔类型、整型、浮点型、字符串、列表、元组、集合、字典等。

1. 空(None)

空表示该值是一个空对象,空值是 Python 中的一个特殊的值,用 None 表示。None 不能理解为 0,因为 0 是有意义的,而 None 是一个特殊的空值。

2. 布尔类型(Boolean)

在 Python 中,None、任何数值类型中的 0、空字符串" "、空元组()、空列表[]、空字典{}都被当作 False。

3. 整型(Int)

整数分为普通整数和长整数,普通整数长度为机器位长,通常都是 32 位,超过这个范围的整数就自动作为长整数处理,而长整数的范围几乎完全没限制。

4. 浮点型(Float)

Python 的浮点数就是数学中的小数,类似 C 语言中的 double。可以用数学写法,如 3.14、-9.01 等。对于很大或很小的浮点数,就必须用科学记数法表示,把 10 用 e 替代。例如,1.23×10^9 就是 1.23e9,0.000012 可以写成 1.2e-5。

2.2.2 变量

Python 是一种动态类型语言,变量不需要显式声明。Python 认为任何数据都是"对象",对变量赋值就是把对象和变量关联起来,变量名是对对象数据的引用,多个对象可以指向同一个变量,每次变量重新赋值,并没有改变对象的值,只是新创建了一个新对象,并用变量指向它,从变量到对象的连接称为引用。

2.2.3 运算符

Python 具有丰富的运算符,如算术运算符、字符串运算符、关系运算符、逻辑运算符、身份运算符、位运算符等。

1. 算术运算符

算术运算符用于处理四则运算。表 2.1 列出了 Python 中的算术运算符。

表 2.1 算术运算符

运算符	描 述	实 例
/	除	10.0/3.0 得到 3.333…3
//	取整除,用于得到商的整数部分	9//2 输出结果 4,9.0//2.0 输出结果 4.0
%	取模运算,返回除法的余数	10%3 得到 1

2. 关系运算符

关系运算符又称比较运算符,是双目运算符,作用是对两个操作数的大小进行比较,比较的结果是一个布尔值,即 True(真)或 False(假)。操作数可以是数值型或字符型。表 2.2 列出了 Python 中的关系运算符。

表 2.2 关系运算符

运算符	描 述	实 例
==	等于	"ABCDE"=="ABR" 返回 False
>	大于	"ABCDE">"ABR" 返回 False
>=	大于或等于	"bc">="大小"返回 False

续表

运算符	描　述	实　例
<	小于	23<3 返回 False
<=	小于或等于	"23"<="3" 返回 True
!=	不等于	"abc"!="ABC" 返回 True

3. 表达式

1）表达式组成

表达式是由数字、运算符和变量等以能求得数值的有意义排列方法所得的组合，表达式通常由运算符（操作符）和参与运算的数（操作数）两部分组成，经过运算后产生的运算结果类型由数据和运算符共同决定。

2）优先级

在一个表达式中，Python 会首先执行优先级高的运算，然后执行优先级低的运算。Python 运算符的优先级在表 2.3 中从上向下依次降低。

表 2.3　运算符的优先级

运　算　符	描　述	运　算　符	描　述
lambda	Lambda 表达式	*,/,%	乘法、除法与取余
or	布尔“或”	+x,-x	正负号
and	布尔“与”	~x	按位翻转
not x	布尔“非”	**	指数
in,not in	成员测试	x.attribute	属性参考
is,is not	同一性测试	x[index]	下标
<,<=,>,>=,!=,==	比较	x[index:index]	寻址段
|	按位或	f(arguments,…)	函数调用
^	按位异或	(experession,…)	绑定或元组显示
&	按位与	[expression,…]	列表显示
<<,>>	移位	{key:datum,…}	字典显示
+,-	加法与减法	'expression,…'	字符串转换

3）结合性

运算符通常由左向右结合，即具有相同优先级的运算符按照从左向右的顺序计算。例如，2+3+4 的计算顺序为(2+3)+4。但是，也有个别运算符从右向左进行计算。例

如，赋值运算符是从右向左结合，即 a=b=c 被处理为 a=(b=c)。

2.3 课后习题答案

一、选择题

1. 下面属于合法的变量名的是（ A ）。

A. X_yz　　B. 123abc　　C. and　　D. X-Y

2. 下面属于不合法的整常数的是（ D ）。

A. 100　　B. &O100　　C. &H100　　D. %100

3. 表达式 16/4-2**5*8/4%5//2 的值为（ D ）。

A. 14　　B. 4　　C. 20　　D. 2

4. 数学关系表达式 3≤x<10 表示成正确的 Python 表达式为（ C ）。

A. 3<=x<10　　B. 3<=x and x<10

C. x>=3 or x<10　　D. 3<=x and <10

5. 与数学表达式 ab/(3cd)对应，Python 的不正确表达式是（ D ）。

A. a*b/(3*c*d)　　B. a/3*b/c/d

C. a*b/3/c/d　　D. a*b/3*c*d

二、用 Python 表达式表示下列命题。

1. n 是 m 的倍数。

【解答】 n % m ==0

2. n 是小于正整数 k 的偶数。

【解答】 n % 2 ==0 and n <int(k)

3. x≥y 或 x<y。

【解答】 x>=y or x<y

4. x、y 中有一个小于 z。

【解答】 y<z or x<z

5. x、y 都小于 z。

【解答】 y<z and x<z

6. x、y 两者都大于 z，且为 z 的倍数。

【解答】 y > z and x > z and y % z ==0 and x % z ==0

三、计算题

设 a=7，b=-2，c=4，求下列表达式的值。

(1) a/2*b/2-c=-7

(2) a % 3+b*b-c//5=5

(3) 1234.5678*a+0.5/100-b=8643.9796

2.4 习题与解答

2.4.1 习题

1. 5%3+3//5 * 2 的运算结果是________。

2. int(1234.5678 * 10+0.5)%100 的运算结果是________。

3. X、Y、Z 表示三角形的 3 条边，条件“三角形任意两边和大于第三边”的布尔表达式可以用________表示。

4. X 是实数，对 X 小数点后第 3 位进行四舍五入的表达式是________。

2.4.2 习题参考答案

1. 5%3+3//5 * 2 的运算结果是________。

【解析】 表达式 5%3+3//5 * 2 中的乘法和除法运算的优先级最高且属同一级运算，因此，先计算 5%3，结果为 2，3//5，结果为 0，此时表达式简化为 2+0 * 2，其运算结果为 2。

【答案】 2

2. int(1234.5678 * 10+0.5)%100 的运算结果是________。

【解答】 本题主要考查 int 函数的功能。int 函数的格式为 int(X)，其功能是求不大于 X 的最大整数，如 int(5.7)，结果为 5，int(－10.9)，结果为－11。对于表达式 int(1234.5678 * 10+0.5) %100，要先计算括号内的参数值，结果为 12346.178，这样整个表达式简化为 int(12346.178)%100。即，int 函数 int(12346.178)的运算结果为 12346，整个表达式结果为 46。

【答案】 46

3. 用 X、Y、Z 表示三角形的 3 条边，条件“三角形任意两边和大于第三边”的布尔表达式可以用________表示。

【解析】 根据“三角形任意两遍之和大于第三边”定义，可得：X+Y>Z，X+Z>Y，Y+Z>X 这 3 个条件必须同时满足。

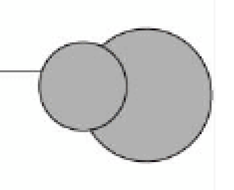
【答案】 X+Y>Z and X+Z>Y and Y+Z>X

4. X 是实数，对 X 小数点后第 3 位进行四舍五入的表达式是________。

【解答】 不妨将 X 取值为 3.246 和 3.243 带入 0.01 * int(100 * (X+0.005))。

【答案】 0.01 * int(100 * (X+0.005))

第3章 顺序与选择结构



3.1 本章要求

- 了解程序设计过程。
- 掌握程序流程图。
- 了解代码书写规则。
- 掌握顺序结构。
- 掌握选择结构。
- 了解程序设计方法与风格。

3.2 本章知识重点

3.2.1 3 种基本逻辑结构

程序处理流程具有输入、处理、输出 3 大步骤，如图 3.1 所示。输入包括变量赋值语句等，处理包括算法、逻辑、计算等，输出包括打印、写入文件或数据库等。

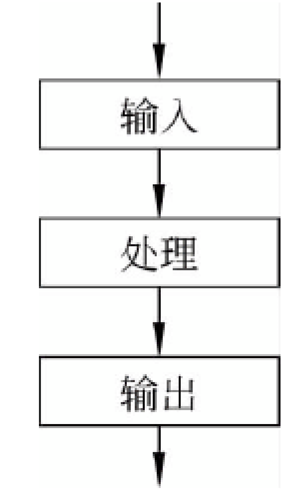


图 3.1 程序处理流程

1996 年，意大利人 Bobra 和 Jacopini 提出顺序结构、选择结构和循环结构 3 种基本结构。顺序结构作为最简单的控制结构，是按照语句书写的先后次序一句一句地执行。选择结构又称为分支语句、条件判定结构，表示在某种特定的条件下选择程序中的特定语句执行，即对不同的问题采用不同的处理方法。循环结构是指只要条件表达式为真，程序就反复、有规律地执行某一操作块的现象。

3.2.2 Python 程序设计流程

采用 Python 设计程序一般分为如下步骤。

步骤 1：分析找出解决问题的关键之处，即找出解决问题的算法，确定算法的步骤。

步骤 2：将算法转换为程序流程图。

步骤 3：根据程序流程图编写符合 Python 语法的代码。

步骤 4：调试程序，纠正错误。

3.2.3 代码书写缩进

Python 语言使用代码书写缩进体现程序的逻辑关系，行首的空白称为缩进，缩进结束就表示一个代码块结束。

(1) Python 语言将代码书写缩进作为语法要求。

(2) Python 利用行首的空白(空格和制表符(Tab 键))来决定逻辑行的缩进层次。

(3) 同一层次的语句必须有相同的缩进，每一组这样的语句称为一个块。

注意： 缩进要么都使用空格，要么都使用制表符，不要混用。

3.2.4 输入与输出

1. 数据输入

Python 提供 raw_input()和 input()两个函数实现数据输入。raw_input()接收字符串类型的数据。input()要求数据必须是数值类型。

2. 数据输出

Python 通过 print 方法实现数据的输出操作，print 方法的语法结构如下：

```
print <expression>,<expression>
```

print 的操作对象是字符串。

【例 3-1】 print 举例。

```
>>>a=5;b="i am a teacher";c=2.5
>>>print("%d,%s,%f"%(a,b,c))
    5,i am a teacher,2.500000
```

3.2.5 顺序结构

顺序结构是最简单的控制结构，按照语句书写先后次序依次执行。顺序结构的语句主要是赋值语句、输入与输出语句等，其特点是程序沿着一个方向进行，具有唯一的入口和出口。如图 3.2 所示，只有先执行语句 1，才会执行语句 2，语句 1 的输出结果作为语句 2 的输入。也就是说，如果没有执行语句 1，语句 2 不会执行。

语句1

语句2

图 3.2 顺序结构

3.2.6 选择结构

选择结构又称为分支结构，表示在某种特定的条件下选择特定语句执行。Python 通过 if 语句来实现分支结构。

if 语句具有单分支、双分支和多分支 3 种形式。

1. 单分支

if 的单分支流程图如图 3.3 所示。其书写格式如下：

```
if 条件表达式：
    语句块
```

2. 双分支

if 语句的双分支流程图如图 3.4 所示。当条件表达式的值为 True(真)时，程序执行语句 1；当条件表达式的值为 False(假)时，程序执行语句 2。

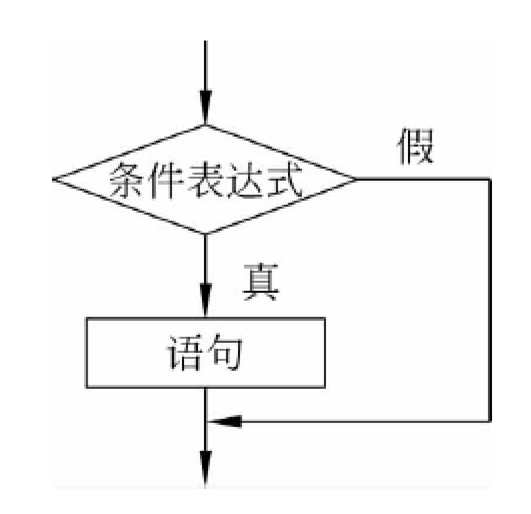


图 3.3 if 的单分支流程图

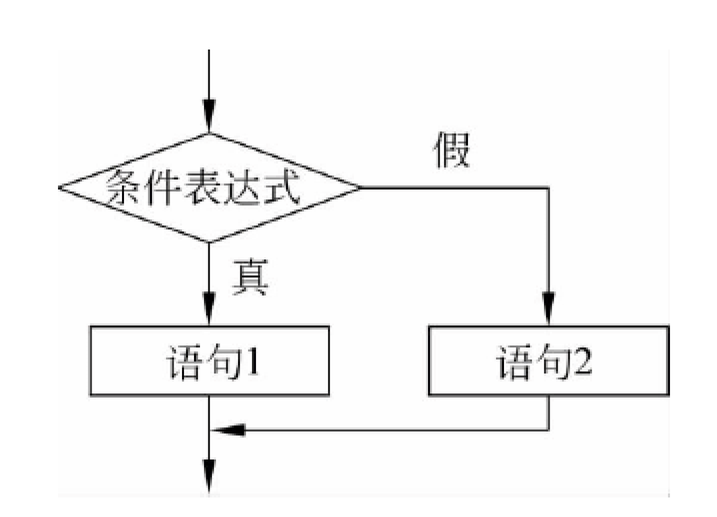


图 3.4 if 语句的双分支流程图

if 的双分支语句书写格式如下：

```
if 条件表达式：
    <语句块 1>
else:
    <语句块 2>
```

3. 多分支

当分支超过两个时，采用 if 语句的多分支语句。该语句的作用是根据不同的条件表达式的值确定执行哪个语句块，Python 测试条件依次为条件表达式 1、条件表达式 2……当某个条件表达式值为 True 时，就执行该条件下的语句块，其余分支不再执行；若所有条件都不满足，且有 else 子句，则执行该语句块，否则什么也不执行。

if 的多分支书写格式如下：

```
if 条件表达式 1:
    <语句块 1>
elif 条件表达式 2:
    <语句块 2>
elif 条件表达式 3:
    <语句块 3>
⋮
else:
    <语句块 n>
```

if 语句的多分支流程图如图 3.5 所示。

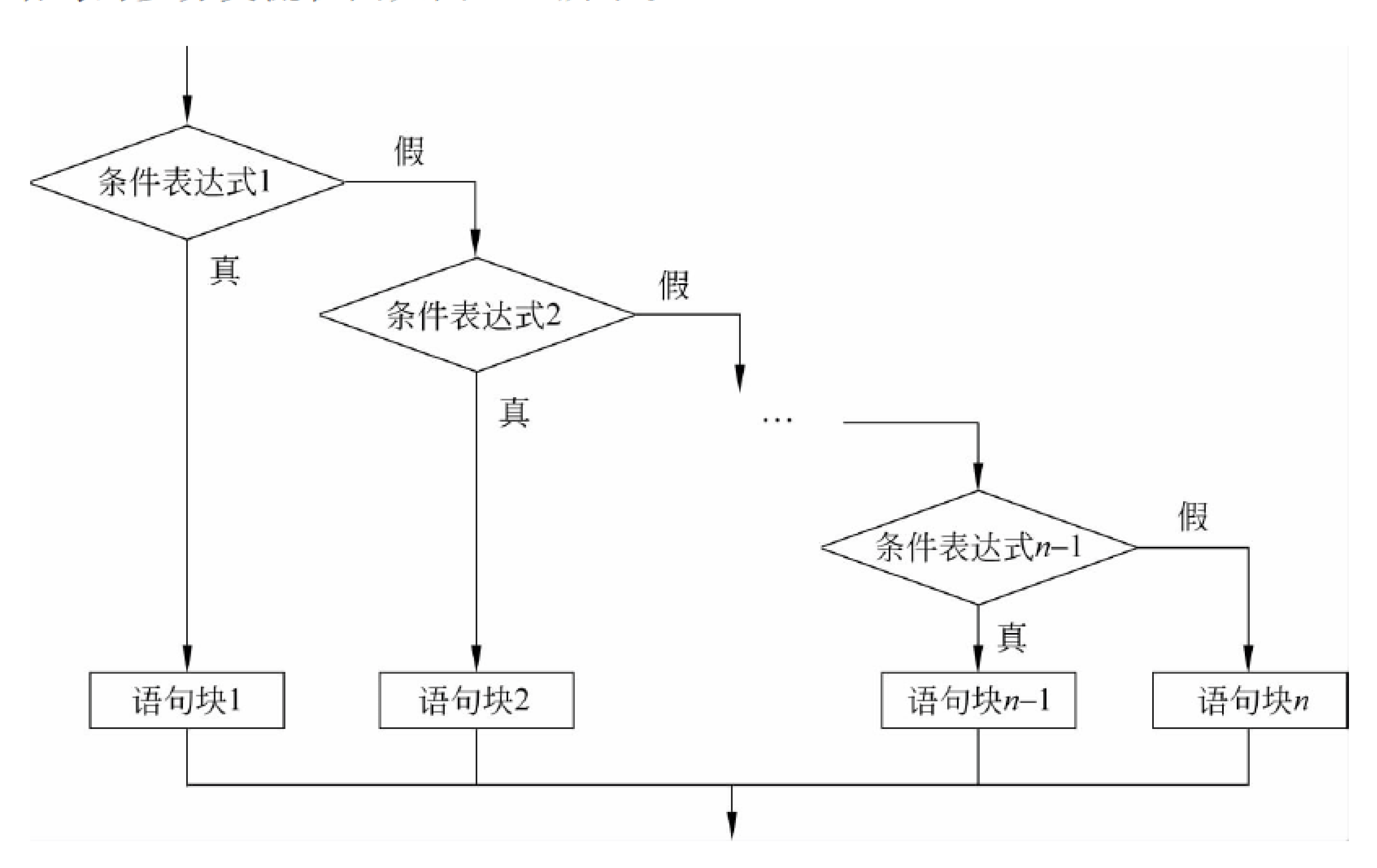


图 3.5　if 语句的多分支流程图

3.3　课后习题答案

一、选择题

1. 在一个语句内写多条语句时，每个语句之间用（ B ）符号分隔。

   A. ，　　B. ；　　C. 、　　D. &

2. 一语句要在下一行继续写，用（ A ）符号作为续行符。

   A. \　　B. -　　C. _　　D. ；

二、编程题

1. 编写一个程序：从键盘输入某个分钟数，将其转化为用小时和分钟表示。

【解答】

```
num= input("please input a number")
hour= num // 60
min = num %60
print "hour is %.f" %hour
print "min is %.f" %min
```

运行结果：

```
please input a number366
hour is 6
min is 6
```

2. 在购买某物品时，若标明的价钱 x 在下式的某个范围内，所付金额 y 按对应折扣

支付，其数学表达式如下：

$$y=\begin{cases}x, & x<1000\\0.9x, & 1000\leqslant x<2000\\0.8x, & 2000\leqslant x<3000\\0.7x, & x>3000\end{cases}$$

编程实现该数学表达式。

【解答】

```
x= input("please input a number")
if x< 1000:
    y= x
elif x< 2000:
    y= 0.9* x
elif x< 3000:
    y= 0.8* x
else:
    y= 0.7* x
print y
```

3. 编写一个程序，判断用户输入的字符是数字字符、字母字符还是其他字符。

【解答】

```
import string
def main():
    c = raw_input('input a string:')
    if c.isalpha():
        print 'letter'
    elif c.isdigit():
        print 'number'
    else:
        print 'other'
if __name__ == '__main__':
    main()
```

运行结果：

```
input a string:5
number
```

4. 编写计算圆面积和球体积的程序。要求输出结果只保留 4 位小数。如果半径的输入值不合法，如含有非数值字符，则提示错误。

【解答】

```
PI= 3.1415926
r= raw_input("please input a number: ")
if r.isdigit():
```

```
        r=int(r)
        area=r*r*PI
        volume=(4/3)*PI*r*r*r
        print 'area is %.4f' %area
        print 'volume is %.4f' %volume
    else:
        print "wrong"
```

运行结果：

```
please input a number:4
area is 50.2655
volume is 201.0619
```

3.4 习题与解答

3.4.1 习题

1. 输入平面上的两个点，计算两点的距离。

2. 将百分制转换为五级制，即成绩不低于90分用"优秀"表示，80～89分用"良好"表示，70～79分用"中等"表示，60～69分用"及格"表示，不低于60分用"不及格"表示。

3. 输入一行字符，分别统计出其中英文字母、空格、数字和其他字符的个数。

4. 输入三角形的3条边，判断能否组成三角形。若能，计算三角形的面积。

3.4.2 习题参考答案

1. 输入平面上的两个点，计算两点的距离。

【解答】

```
import math
x1,y1=input('please the start point x1,y1:')
x2,y2=input('please the start point x2,y2:')
distance=math.sqrt((x1-x2)**2+ (y1-y2)**2)
print'distance=',distance
```

运行结果：

```
please the start point x1,y1:0,0
please the start point x2,y2:3,4
distance=5.0
```

2. 将百分制转换为五级制，即成绩不低于90分用"优秀"表示，80～89分用"良好"表示，70～79分用"中等"表示，60～69分用"及格"表示，不低于60分用"不及格"表示。

【解答】

```
def main():
```

```
    s = int(raw_input('Enter a number:'))
    if s>=90:
        grade = 'A'
    elif s>=80:
        grade = 'B'
    elif s>=70:
        grade = 'C'
    elif s>=60:
        grade = 'D'
    else:
        grade = 'E'
    print grade

if __name__ == '__main__':
    main()
```

3. 输入一行字符，分别统计出其中英文字母、空格、数字和其他字符的个数。

【解答】

```
import string
def main():
    s = raw_input('input a string:')
    letter = 0
    space = 0
    digit = 0
    other = 0
    for c in s:
        if c.isalpha():
            letter+=1
        elif c.isspace():
            space+=1
        elif c.isdigit():
            digit+=1
        else:
            other+=1
    print 'There are %d letters,%d spaces,%d digits and %d other characters in your string.'% (letter,
    space,digit,other)
if __name__ == '__main__':
    main()
```

运行结果：

```
input a string:SF4D **2JR8F nnD992&.&.
There are 9 letters,3 spaces,6 digits and 8 other characters in your string.
```

4. 输入三角形的 3 条边，判断能否组成三角形。若能，计算三角形的面积。

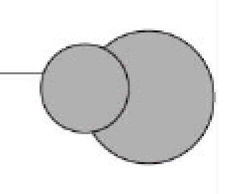
【解答】

```
import math
a,b,c= input('please input 3 numbers(","):')
if a+b> c and a+c> b and b+c> a:
    print 'this is a triangle !'
    p= (a+b+c)/2
    areaspr=p* (p-a)* (p-b)* (p-c)
    area=math.sqrt(areaspr)
    print 'area= ',area
else:
    print'that is not triangle!'
```

运行结果：

```
please input 3 numbers(","):3,4,5
this is a triangle !
area= 6.0
```

第4章 循环结构



4.1 本章要求

- 了解循环结构的概念。
- 掌握 while 语句。
- 掌握 for 语句。
- 掌握辅助语句。
- 掌握循环嵌套。

4.2 本章知识重点

4.2.1 构造循环结构

循环是指程序从某处开始有规律地反复执行某一操作块。循环由循环体及循环控制条件两部分组成。反复执行的语句或程序段称为循环体。循环体是否能继续执行，取决于循环控制条件的真假。图 4.1 给出了构造循环的流程图。

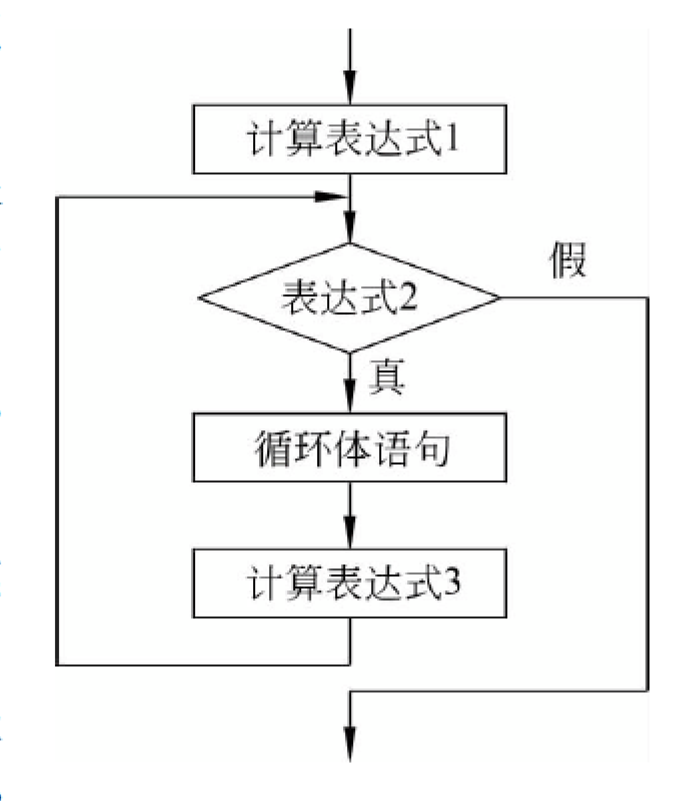


图 4.1 构造循环结构

构造循环结构的关键是确定与循环控制变量(i)有关的 3 个表达式：表达式 1、表达式 2 和表达式 3，下面详细介绍。

表达式 1 为循环控制变量初值，作为循环开始的初始条件，给循环控制变量赋初始值。

表达式 2 用于判断是否执行循环体。当满足表达式 2 时，循环体反复被执行；反之，当表达式 2 为假时，退出循环体，不再执行。设想，如果表达式 2 始终为真，循环体将会一直被执行，成为“死循环”。那么如何终止循环呢？也就是说，如何让表达式 2 为假？为此产生了表达式 3。

表达式 3 是关于循环控制变量的某种运算。循环体每执行一次，表达式 3 也执行一次，循环控制变量的改变最终导致表达式 2 结果为假，循环终止。

Python 语言中，循环结构有 while 和 for 两种。

4.2.2 while 语句

while 语句有如下两种书写格式:

【格式一】

```
while 循环控制条件:
    循环体
```

【格式二】

```
while 循环控制条件:
    循环体
else:
    语句
```

【例 4-1】 计算 1～100 的所有整数之和。

【解析】 计算一批数据的和称为累加。通常引入变量 sum 存放“部分和”,变量 i 存放“累加项”,用于表示变化的量。通过执行“部分和=部分和+累加项”实现累加。

```
i=1                        #表达式 1,i 为循环变量
sum=0                      #sum表示累加的和
while i<=100:              #表达式 2,i 的变化范围为 1~100
    sum = sum + i          #部分和累加
    i+=1                   #表达式 3,i 的步长为 1
print "sum=",sum           #总和
```

运行结果:

```
sum=5050
```

4.2.3 for 语句

for 语句是遍历型循环,依次访问序列中的全体元素,主要用于列表、元组等迭代结构。

for 语句书写格式如下:

```
for 目标标识符 in 序列:
    循环体
```

【例 4-2】 for 循环应用于列表序列。

```
fruits = ['banana', 'apple', 'mango']      #列表
for fruit in fruits:
    print 'fruits have :', fruit
```

输出如图 4.2 所示。

4.2.4 循环嵌套

一个循环体中嵌入另一个循环,这种情况称为多重循环,又称循环嵌套,较常见的是



图 4.2　例 4-2 程序运行结果

二重循环。Python 语言允许在 while 循环中嵌入 for 循环,反之亦可。

Python 循环嵌套语法如下所示:

```
while expression:
    for iterating_var in sequence:
        statement(s)
    statement(s)
```

二重循环结构的构造需要确定外层循环控制变量、内层循环控制变量以及内外层循环控制变量之间的关系。一般具有如下步骤。

步骤 1:确定其中一个循环控制变量为定值,实现单重循环。

步骤 2:将此循环控制变量从定值变化成变值,将单重循环转变为双重循环。

【例 4-3】 打印九九乘法表。

【解析】 九九乘法表涉及乘数 i 和被乘数 j 两个变量,变化范围为 1～9。先假设被乘数 j 的值不变,假设为 1,实现单重循环。

```
for i in range(1,10):
    j = 1
    print i ,"* " ,j , "=" ,i * j
```

程序运行结果如图 4.3 所示。

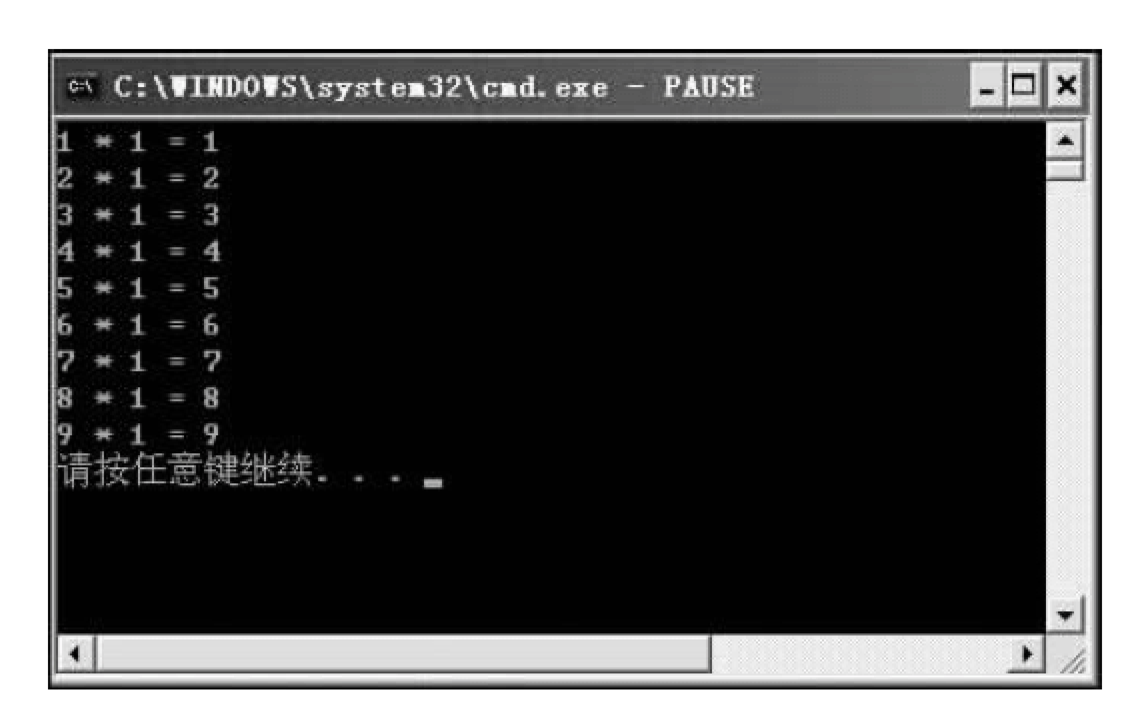


图 4.3　例 4-3 程序运行结果一

【解析】 将被乘数 j 的定值 1 改为变量,让其在 1～9 之间取值。

```
for i in range(1,10):
```

```
    for j in range(1,10):
        print i ,"* " ,j , "=" ,i * j
    print
```

程序运行结果如图 4.4 所示。

```
C:\WINDOWS\system32\cmd.exe - PAUSE
1 * 1 = 1 1 * 2 = 2 1 * 3 = 3 1 * 4 = 4 1 * 5 = 5 1 * 6 = 6 1 * 7 = 7 1 * 8 = 8 1 * 9 = 9
2 * 1 = 2 2 * 2 = 4 2 * 3 = 6 2 * 4 = 8 2 * 5 = 10 2 * 6 = 12 2 * 7 = 14 2 * 8 = 16 2 * 9 = 18
3 * 1 = 3 3 * 2 = 6 3 * 3 = 9 3 * 4 = 12 3 * 5 = 15 3 * 6 = 18 3 * 7 = 21 3 * 8 = 24 3 * 9 = 27
4 * 1 = 4 4 * 2 = 8 4 * 3 = 12 4 * 4 = 16 4 * 5 = 20 4 * 6 = 24 4 * 7 = 28 4 * 8 = 32 4 * 9 = 36
5 * 1 = 5 5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25 5 * 6 = 30 5 * 7 = 35 5 * 8 = 40 5 * 9 = 45
6 * 1 = 6 6 * 2 = 12 6 * 3 = 18 6 * 4 = 24 6 * 5 = 30 6 * 6 = 36 6 * 7 = 42 6 * 8 = 48 6 * 9 = 54
7 * 1 = 7 7 * 2 = 14 7 * 3 = 21 7 * 4 = 28 7 * 5 = 35 7 * 6 = 42 7 * 7 = 49 7 * 8 = 56 7 * 9 = 63
8 * 1 = 8 8 * 2 = 16 8 * 3 = 24 8 * 4 = 32 8 * 5 = 40 8 * 6 = 48 8 * 7 = 56 8 * 8 = 64 8 * 9 = 72
9 * 1 = 9 9 * 2 = 18 9 * 3 = 27 9 * 4 = 36 9 * 5 = 45 9 * 6 = 54 9 * 7 = 63 9 * 8 = 72 9 * 9 = 81
请按任意键继续. . .
```

图 4.4　例 4-3 程序运行结果二

4.3　课后习题答案

1. 求 1～100 之间所有的素数，并统计素数的个数。

【解答】　一个大于 1 的自然数，除了 1 和它本身外，不能被其他自然数(质数)整除(2，3，5，7 等)的数称为素数，换句话说就是素数只能被 1 和其自身整除。

```
num=2
counter=0
while num<100:
    flag=1
    i=2
    while i <num:
        if num %i==0:
            flag=0
            break
        i=i+1
    if flag==1:
        counter=counter+1
        print num
    num=num +1
print
print "The counter of prime is %.f" %counter
```

运行结果：

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
The counter of prime is 25
```

2. 求 200 以内能被 17 整除的最大正整数。

【解答】

```
n = 200
while n< = 200:
    if n %17= = 0 :
        print n
        break
    n = n - 1
```

运行结果：

```
187
```

3. 设 $m=1\times2\times3\times\cdots\times n$，求 m 为不大于 20 000 时最大的 n。

【解答】

```
m = 1
n = 1
while m< = 20000:
    m = m * n
    n = n + 1
n = n - 1
print "Max n = %.f" %n
```

运行结果：

```
Max n = 8
```

4. 勾股定理中 3 个数的关系是 $a^2+b^2=c^2$。编写一个程序，输出 30 以内满足上述条件的整数组合，如 3、4、5 就是一个组合。

【解答】

```
for a in range(1,31,1):
    for b in range(1,31,1):
        for c in range(1,31,1):
            if a* a+b* b= = c* c:
                print("%d%d%d"% (a,b,c))
```

运行结果：

```
3,4,5
4,3,5
5,12,13
6,8,10
7,24,25
8,6,10
8,15,17
9,12,15
```

```
10,24,26
12,5,13
12,9,15
12,16,20
15,8,17
15,20,25
16,12,20
18,24,30
20,15,25
20,21,29
21,20,29
24,7,25
24,10,26
24,18,30
```

5. 625 这个数字很特别，625 的平方等于 390 625，刚好其末 3 位是 625 本身。请编写程序，寻找所有这样的 3 位数：它的平方的末 3 位是这个数字本身。

【解答】

```
n=100
while n<=1000:
    m=n*n
    if n==m%1000:
        print n
    n=n+1
```

运行结果：

```
376  625
```

6. 求 1～100 之间能被 7 整除，但不能同时被 5 整除的所有整数。

【解答】

```
n=1
while n<=100:
    if n%7==0 and n%5 !=0:
        print n
    n=n+1
```

运行结果：

```
7 14 21 28 42 49 56 63 77 84 91 98
```

7. 编程得到如图 4.5 所示的输出。

【解答】 第 1 幅图有如下两种方法。

方法 1：由表 4.1 给出的空格数、星号数和行数的关系推出公式 j=7−2*i 和 k=i−1。

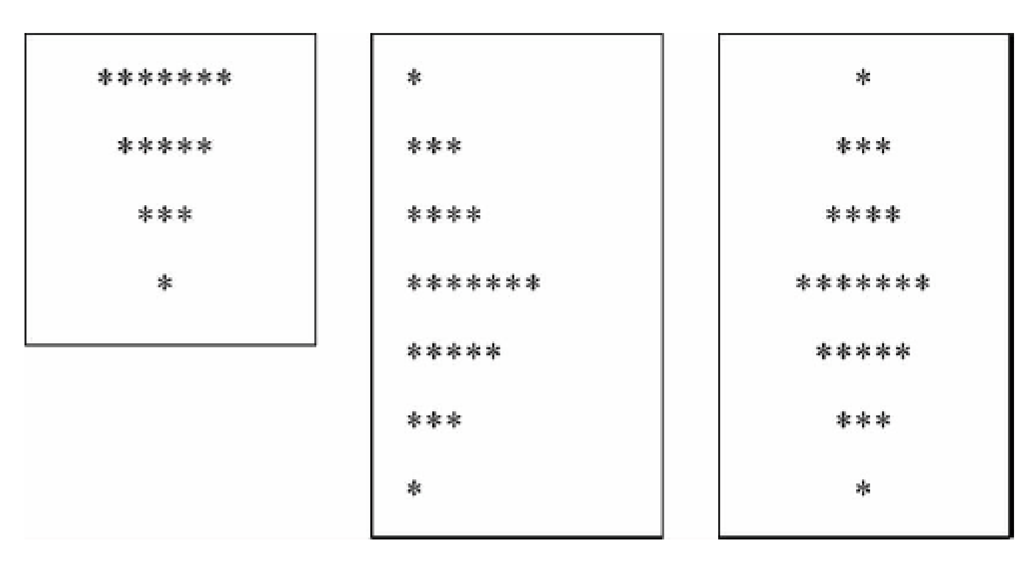


图 4.5　题 7 程序运行结果

表 4.1　空格数、星号数和行数的关系表

I(行数)	K(空格数)	J(星号数)	I(行数)	K(空格数)	J(星号数)
1	0	7	3	2	3
2	1	5	4	3	1

代码如下：

```
for i in range(0,4):
    for k in range(0,i):
        print " "
    for j in range(0,7-2* i):
    print "* "
print
```

方法 2：使用乘号'*'的特性。

代码如下：

```
for i in range(7,0,-2):
    print ' '* (4-(i+1)/2)+'* '* i
```

第 2 幅图可以分为两部分，前 4 行一个规律，后 3 行另一个规律。

代码如下：

```
for i in range(1,8,2):
    print '* '* i
for i in range(7,0,-2):
    print '* '* i
```

第 3 幅图与第 2 幅图有类似之处。

代码如下：

```
for i in range(1,8,2):
    print ' '* (4-(i+1)/2)+'* '* i
for i in range(5,0,-2):
    print ' '* (4-(i+1)/2)+'* '* i
```

4.4 习题与解答

4.4.1 习题

1. 输入若干成绩，计算并输出平均成绩，输入每个成绩后询问是否继续输入成绩。

2. 输入若干正整数，求所有输入的数之和，遇到负数即结束。

3. 输入 n 的值，计算 $s=1+1/2+\cdots+1/n!$。

4. 求 200 以内能够被 13 整除的最大的整数，并输出。

5. 求 1～100 之间所有偶数的和。

6. 猜数游戏。预设一个 0～9 的整数，让用户输入所猜的数。如果大于预设的数，显示“bigger”；如果小于预设的数，显示“smaller”。如此循环，直至猜中该数，显示“right!”。

7. 猴子吃桃问题。猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天又将剩下的桃子吃掉一半，又多吃了一个。以后每天都吃了前一天剩下的一半加一个。到第 10 天，只剩下一个桃子了。求第一天共摘了多少个桃子。

8. 用数字 1、2、3、4 能组成多少个互不相同且无重复数字的三位数？打印结果。

9. 采用简单选择排序法对 list=[2,6,7,3,9,1,4,7,90,34]进行升序排列。

10. 求 1!+2!+…+10!

4.4.2 习题参考答案

1. 输入若干成绩，计算并输出平均成绩，输入每个成绩后均询问是否继续输入成绩。

【解答】

```
sum=0.0
num=0
flag='y'
while flag=='y':
    score=input('please input a score:')
    num=num+1
    sum=sum+score
    average=sum/num
    print'the average is:',average
    flag=raw_input('would you like to go on?(y/n)')
```

运行结果：

```
please input a score:90
the average is: 90.0
would you like to go on?(y/n)y
please input a score:78
the average is: 84.0
would you like to go on?(y/n)y
```

```
please input a score:86
the average is: 84.6666666667
would you like to go on?(y/n)n
```

2. 输入若干正整数,求所有输入的数之和,遇到负数即结束。

【解答】

```
sum=0
num=input('please input a number,a negtive means the end:')
while num>0:
    sum=sum+num
    num=input('please input a number,a negtive means the end:')
print'the sum of num is :',sum
```

运行结果:

```
please input a number,a negtive means the end:5
please input a number,a negtive means the end:7
please input a number,a negtive means the end:6
please input a number,a negtive means the end:-2
the sum of num is: 18
```

3. 输入 n 的值,计算 $s=1+1/2+\cdots+1/n!$。

【解答】

```
num=input('please input a number:')
sum=1.0
sumall=0.0
n=1
while n<=num:
    sum=sum*n
    sumall=sumall+1/sum
    n=n+1
print 'the addition is:',sumall
```

运行结果:

```
please input a number: 2
the addition is: 1.5
```

4. 求 200 以内能够被 13 整除的最大的整数,并输出。

【解答】

```
for i in range(200,0,-1):
    if i%13==0:
        break
print 'the biggest one that can be divided by 13 in 200 is:',i
```

运行结果:

```
the biggest one that can be divided by 13 in 200 is: 195
```

5. 求 1～100 之间所有偶数的和。

【解答】

```
sum=0
for x in range(1,101):
    if x %2==0:
        sum=sum+x
print("1~100 sum is:",sum)
```

运行结果：

```
'1~100 sum is:', 2550
```

6. 猜数游戏。预设一个 0～9 的整数，让用户输入所猜的数。如果大于预设的数，显示“bigger”；如果小于预设的数，显示“smaller”。如此循环，直至猜中该数，显示“right!”。

【解答】

```
num=7
while True:
    guess=int(input("please input a number:"))
    if guess ==num:
        print("right!")
        break;
    elif guess >num:
        print("bigger")
    else:
        print("smaller")
```

运行结果：

```
please input a number:5
smaller
please input a number:6
smaller
please input a number:8
bigger
please input a number:7
right!
```

7. 猴子吃桃问题。猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天又将剩下的桃子吃掉一半，又多吃了一个。以后每天都吃了前一天剩下的一半加一个。到第 10 天，只剩下一个桃子了。求第一天共摘了多少个桃子。

【解答】

```
day=9
```

```
x=1
while day>0:
    x=(x+1)*2
    day-=1
print "total=",x
```

运行结果：

```
total=1534
```

8. 用数字 1、2、3、4 能组成多少个互不相同且无重复数字的三位数？打印结果。

【解答】 可以填在百位、十位、个位的数字是 1、2、3、4，组成所有的排列后再去掉不满足条件的排列。

```
counter=0
for i in range(1,5):
    for j in range(1,5):
        for k in range(1,5):
            if(i!=k) and (i!=j) and (j!=k):
                counter+=1
                print i,j,k
print
print counter
```

运行结果：

```
1 2 3
1 2 4
1 3 2
1 3 4
1 4 2
1 4 3
2 1 3
2 1 4
2 3 1
2 3 4
2 4 1
2 4 3
3 1 2
3 1 4
3 2 1
3 2 4
3 4 1
3 4 2
4 1 2
4 1 3
```

```
4 2 1
4 2 3
4 3 1
4 3 2

24
```

9. 采用简单选择排序法对 list=[2,6,7,3,9,1,4,7,90,34]进行升序排列。

【解答】 采用选择排序法对一组关键字$\{K_1,K_2,\cdots,K_n\}$,首先从$K_1,K_2,\cdots,K_n$中选择最小值,假如它是K_z,则将K_z与K_1对换;然后从$K_2,K_3,\cdots,K_n$中选择最小值K_z,再将K_z与K_2对换……直至只剩下一个关键字K_n,此时一个由小到大的有序序列就排好了。

```
def selection_sort(list):
    for i in range(0, len (list)):
        min = i
        for j in range(i + 1, len(list)):
            if list[j] < list[min]:
                min = j
        list[i], list[min] = list[min], list[i]        #交换

list2= [2,6,7,3,9,1,4,7,90,34]
selection_sort(list)
print list
```

运行结果:

```
[1, 2, 3, 4, 6, 7, 7, 9, 34, 90]
```

10. 求1!+2!+…+10!。

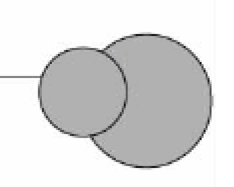
【解答】

```
n= 0;s= 0;t= 1
for n in range(1,11):
    t* =n
    s+=t
print 'sum is %d' %s
```

运行结果:

```
sum is 4037913
```

第5章 序列与字典



5.1 本章要求

- 了解序列。
- 掌握列表。
- 掌握元组。
- 掌握字符串。
- 掌握字典。
- 掌握 JSON。

5.2 本章知识重点

5.2.1 序列

在 Python 中,最基本的数据结构是序列(sequence)。序列中的元素有序排列,每个元素被分配一个序号,即元素的位置,也称为索引。第一个索引是 0,第二个是 1,以此类推。也可以采用负号索引,即序列中的最后一个元素位置标记为−1,倒数第二个元素位置为−2,以此类推。

Python 提供了列表、元组、字符串等序列类型,可以进行某些特定的操作。这些操作包括索引(index)、分片(slice)、加(add)、乘(multiply)以及检查某个元素是否属于序列的成员(成员资格)。除此之外,Python 还有计算序列长度、找出最大元素和最小元素的内建函数。

5.2.2 列表

现实生活中的购物清单、手机通讯录等都可以看作一个列表,列表(list)是一组有序项目的数据结构。Python 创建列表时,解释器在内存中生成一个类似数组的数据结构来存储数据,数据项自下而上存储。

Python 列表可以包含混合类型的数据,即在一个列表中的数据类型可以各不相同。列表中的每一个数据称为元素,元素用逗号分隔并放在一对中括号“[”和“]”中,列表可以认为是下标从 0 开始的数组。

【例 5-1】 列表举例。

```
# coding=UTF-8
animalist = ['fox','tiger','rabbit','snake']
print('Zoo has ',len(animalist),'animals...')
for item in animalist:
    print(item),
print('\n')
animalist.append('pig')
del animalist[0]
animalist.sort()
for i in range(0,len(animalist)):
    print(animalist[i])
```

运行结果：

```
('Zoo has ', 4, 'animals...')
fox tiger rabbit snake
pig rabbit snake tiger
```

5.2.3 元组

元组(tuple)和列表类似，但其元素不可变，即元组一旦创建，用任何方法都不可以修改其元素，因此，元组相当于只读列表。

元组与列表相比有如下相同点：

(1) 元组的元素与列表一样按定义的次序进行排序。

(2) 元组的负数索引与列表一样从尾部开始计数。

(3) 元组与列表一样也可以使用分片。

元组与列表相比有如下不同点：

(1) 元组在定义时所有元素是放在一对圆括号"("和")"中，而不是方括号。

(2) 不能向元组中增加元素，元组没有 append 或 extend 方法。

(3) 不能从元组删除元素，元组没有 remove 或 pop 方法。

(4) 元组没有 index 方法，但是可以使用 in 方法。

(5) 元组可以在字典中被用做"键"，但是列表不行。

元组适合只需进行遍历操作的运算，对于数据进行"写保护"，其操作速度比列表快。

【例 5-2】 元组举例。

```
# coding=UTF-8
animalist = ('fox','tiger','rabbit','snake')
for item in animalist:
    print(item)
print('\n')
animalist.append('pig')
```

运行结果：

```
Traceback (most recent call last):
    File "C:/Users/Administrator/PycharmProjects/keygame/keygame.py", line 9, in <module>
    animalist.append('pig')
AttributeError: 'tuple' object has no attribute 'append'
fox tiger rabbit snake
```

5.2.4　字符串

字符串是用单引号、双引号或者三引号括起来的符号系列，例如以'或"括起来的任意文本，如'abc'、"xyz"等。请注意，'或"本身只是一种表示方式，不是字符串的一部分，因此，字符串'abc'只有 a、b、c 这 3 个字符。

字符串方法如表 5.1 所示。

表 5.1　字符串方法

函　　数	描　　述
s. index(sub[,start, end])	
s. find(sub[,start,end]})	与 index 函数一样，但如果找不到会返回－1
s. replace(old, new [,count])	替换 s 里所有 old 子串为 new 子串，count 指定多少个可被替换
s. count(sub[,start,end])	
s. split()	字符串的 split 函数默认分隔符是空格。如果没有分隔符，就把整个字符串作为列表的一个元素
s. join()	join()方法是 split()方法的逆方法，用来把字符串连接起来
s. lower()	返回将大写字母变成小写字母的字符串
s. upper()	返回将小写字母变成大写字母的字符串

5.2.5　字典

字典中的元素由一对称为键和值的项构成，元素的键和值之间用冒号分隔，元素之间用逗号分隔，整体用一对大括号“{”和“}”括起来。字典语法结构如下：

```
dict_name= {key1:value1,key2:value2,… }
```

对于字典，应注意如下几点：

(1) 键必须是唯一的。

(2) 键和值之间用冒号分隔，而各个元素之间用逗号分隔，所有这些都包括在大括号中。

(3) 元素没有顺序。

【例 5-3】　字典举例。

```
# coding=utf-8
dict1 = {'zhang':"张辉",'wang':"王强",'li':"李冰",'zhao':"赵薇"}
dict1['huang'] = '黄家驹'
del dict1['zhao']
for firstname,name in dict1.items():
    print firstname,name
```

运行结果:

```
li 李冰
wang 王强
huang 黄家驹
zhang 张辉
```

5.2.6 JSON

JSON(JavaScript Object Notation,JavaScript 对象表示法)是一种轻量级的数据交换格式。JSON 不但易于人阅读和编写,而且易于计算机解析和生成。其书写格式为"名称:值",例如"firstName":"John"。

Python 2.6 以上版本自带 JSON 模块,具有序列化(encoding)与反序列化(decoding)。

1. 序列化

序列化是指将 Python 对象编码转换成 JSON 字符串,采用 json.dumps 方法。

【例 5-4】 序列化举例。

```
>>> import json
>>> data = [{'a':"A",'b':(2,4),'c':3.0}]
>>> print "DATA:",repr(data)
DATA: [{'a': 'A', 'c': 3.0, 'b': (2, 4)}]
>>> data_string = json.dumps(data)
>>> print "JSON:",data_string
JSON: [{"a": "A", "c": 3.0, "b": [2, 4]}]
```

2. 反序列化

反序列化是指将 JSON 格式字符串解码转换成 Python 对象,采用 json.loads 方法。

【例 5-5】 反序列化举例。

```
>>> import json
>>> data = [{'a':"A",'b':(2,4),'c':3.0}]
>>> data_string = json.dumps(data)
>>> print "ENCODED:",data_string
ENCODED: [{"a": "A", "c": 3.0, "b": [2, 4]}]
```

```
>>>decoded = json.loads(data_string)
>>>print "DECODED:",decoded
DECODED: [{u'a': u'A', u'c': 3.0, u'b': [2, 4]}]
>>>print "ORIGINAL:",type(data[0]['b'])
ORIGINAL: <type 'tuple'>
>>>print "DECODED:",type(decoded[0]['b'])
DECODED: <type 'list'>
```

5.3 课后习题答案

1. 输入一段英文，求其字符串长度，并求出其中包含多少个单词。

【解答】

```
s= raw_input("please input a string:")
len= len(s)
counter= 0
for i in s.split(' '):
    if i:
        counter+ = 1
print "The length is:%.f"%len
print "The counter is:%.f"%counter
```

运行结果：

```
please input a string:I am a boy
The length is:10
The counter is:4
```

2. 输入 10 个成绩，进行优、良、中、及格、不及格的统计。

【解答】

```
counter= 0
exce= 0;good= 0;normal= 0;pas= 0;bad= 0
while counter< = 10:
    number= input("please input a score:")
    if number> = 90:
        exce+ = 1
    elif number> = 80:
        good+ = 1
    elif number> = 70:
        normal+ = 1
    elif number> = 60:
        pas+ = 1
    else:
        bad+ = 1
```

```
    counter+=1
print "exce is:%.f"%exce
print "good is:%.f"%good
print "normal is:%.f"%normal
print "pas is:%.f"%pas
print "bad is:%.f"%bad
```

运行结果：

```
please input a score:67
please input a score:45
please input a score:90
please input a score:44
please input a score:88
please input a score:97
please input a score:36
please input a score:98
please input a score:78
please input a score:67
exce is:3
good is:1
normal is:1
pas is:2
bad is:3
```

3. 输入 10 个学生的姓名和成绩构成的字典，按照成绩大小排序。

【解答】

```
studscore = {}
counter= 0
while counter< 10:
    key = raw_input('Input name:')
    value = raw_input('Input score:')
    studscore[key] = value
    counter+ = 1
dict= sorted(studscore.iteritems(),key= lambda d:d[1])
print "order by score"
print dict
```

运行结果：

```
Input name:wang
Input score:90
Input name:zhang
Input score:67
Input name:hai
Input score:45
```

```
Input name:zhou
Input score:78
Input name:jin
Input score:89
Input name:pan
Input score:87
Input name:shui
Input score:78
Input name:tai
Input score:67
Input name:tian
Input score:67
Input name:ff
Input score:56
order by score
[('hai', '45'), ('ff', '56'), ('zhang', '67'), ('tian', '67'), ('tai', '67'),
('zhou', '78'), ('shui', '78'), ('pan', '87'), ('jin', '89'), ('wang', '90')]
```

4. 输入 10 个学生的姓名和年龄构成的字典，读出其键和值，并分别保存到两个列表中。

【解答】

```
>>> studscore = {'a': 45, 'b': 78, 'c': 40, 'd': 96, 'e': 65, 'f': 90, 'g': 78,
'h': 99, 'i': 60, 'j': 87}
>>> studscore.values()
[45, 40, 78, 65, 96, 78, 90, 60, 99, 87]
>>> studscore.keys()
['a', 'c', 'b', 'e', 'd', 'g', 'f', 'i', 'h', 'j']
```

5. 任意输入一串字符，输出其中的不同字符及其个数。例如，输入 abcdefgabc，输出为 a—>2，b—>2，c—>2，d—>1，e—>1，f—>1，g—>1。

【解答】

```
# !/usr/bin/env python
# coding=utf-8
s=raw_input("please input string: ")
ms = set(s)
for item in ms:
    print item,'->',s.count(item)
```

运行结果：

```
please input string: abcdcbxdcbaxbcc
a->2
x->2
c->5
```

```
b->4
d->2
```

5.4 习题与解答

5.4.1 习题

1. 在列表中输入多个数据作为圆的半径，得出相应的圆的面积。

2. 将 3 行 3 列的矩阵按其形状输出，例如，输入为[[1,2,3],[4,5,6],[7,8,9]]，输出为

```
1    2    3
4    5    6
7    8    9
```

3. 已知列表 b1=[1,2,3]和 b2=[2,3,4]，求 b1 和 b2 的交集和差集。

4. 实现 Python 中的字典和 JSON 互转操作。

5.4.2 习题参考答案

1. 在列表中输入多个数据作为圆的半径，得出相应的圆的面积。

【解答】

```
radius=input('please input radiuses in list')
for r in radius:
    print 'the area of the circle with the radius of %f is:' %r,3.14*r*r
```

运行结果：

```
please input radiuses in list[2,3,4]
the area of the circle with the radius of 2.000000 is: 12.56
the area of the circle with the radius of 3.000000 is: 28.26
the area of the circle with the radius of 4.000000 is: 50.24
```

2. 将 3 行 3 列的矩阵按其形状输出，例如，输入为[[1,2,3],[4,5,6],[7,8,9]]，输出为

【解答】

```
a=input('please input a 3*3 array:')
for x in a:
    s=''
    for y in x:
        s1='%6d'%y
        s=s+s1
    print s
```

运行结果：

```
please input a 3* 3 array:[[1,2,3],[4,5,6],[7,8,9]]
    1     2     3
    4     5     6
    7     8     9
```

3．已知列表 b1=[1,2,3]和 b2=[2,3,4]，求 b1 和 b2 的交集和差集。

【解答】

（1）交集：

```
b1= [1,2,3]
b2= [2,3,4]
b3 = [val for val in b1 if val in b2]
print b3
```

运行结果：

```
[2, 3]
```

（2）差集：

```
b1= [1,2,3]
b2= [2,3,4]
b3= [val for val in b1 if val not in b2]
print b3
```

运行结果：

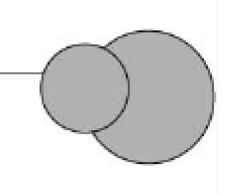
```
[1]
```

4．实现 Python 中的字典和 JSON 互转操作。

【解答】 Python 的字典和 JSON 在表现形式上非常相似，实际上 JSON 就是 Python 字典的字符串表示。

```
>>> import json
>>> stu = '{"name": "zhang san", "sex": "male", "age": 25}'
>>>b = json.loads(stu)
>>>b.values()
[25, 'zhang san', 'male']
```

第 6 章



数据结构与算法

6.1 本章要求

- 掌握数据结构的概念。
- 了解算法。

6.2 本章知识重点

6.2.1 数据结构

数据结构研究各种相关的数据信息如何表示、组织、存储与加工处理。数据结构中的关系指数据间的逻辑关系，与数据的物理存储无关，是从具体问题抽象出来的数学模型。数据结构一般有线性结构和非线性结构。

1. 线性结构

线性结构是指元素与元素之间是一对一的关系，一般有线性表、栈和队列等结构。

1) 线性表

线性表$(a_0, a_1, \cdots, a_n)(n>0)$如图 6.1 所示，具有如下特点：

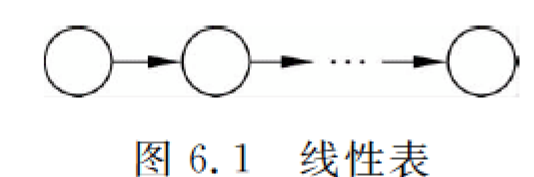


图 6.1 线性表

- 存在唯一的“第一元素”a_0。
- 存在唯一的“最后元素”a_n。
- 除最后元素 a_n 之外，各元素均有唯一的后继。
- 除第一元素 a_0 之外，各元素均有唯一的前驱。

2) 栈和队列

从数据结构角度讲，栈和队列均是操作受限的线性表，两者不同之处在于操作的特殊性(栈为 LIFO，即后进先出；队列为 FIFO，即先进后出)。

栈是限定仅在表尾进行插入或删除操作的线性表，具有后进先出的特性，即最先进入的元素最后一个被释放。栈的逻辑结构如图 6.2 所示。

栈具有如下两个操作：

- 入栈(PUSH)，最先插入的元素放在栈的底部。

- 出栈(POP),最后插入的元素最先出栈。

队列只允许在线性表的一端(队尾)进行插入(入队列),而在另一端(队头)进行删除(出队列)。队列的逻辑结构如图 6.3 所示。

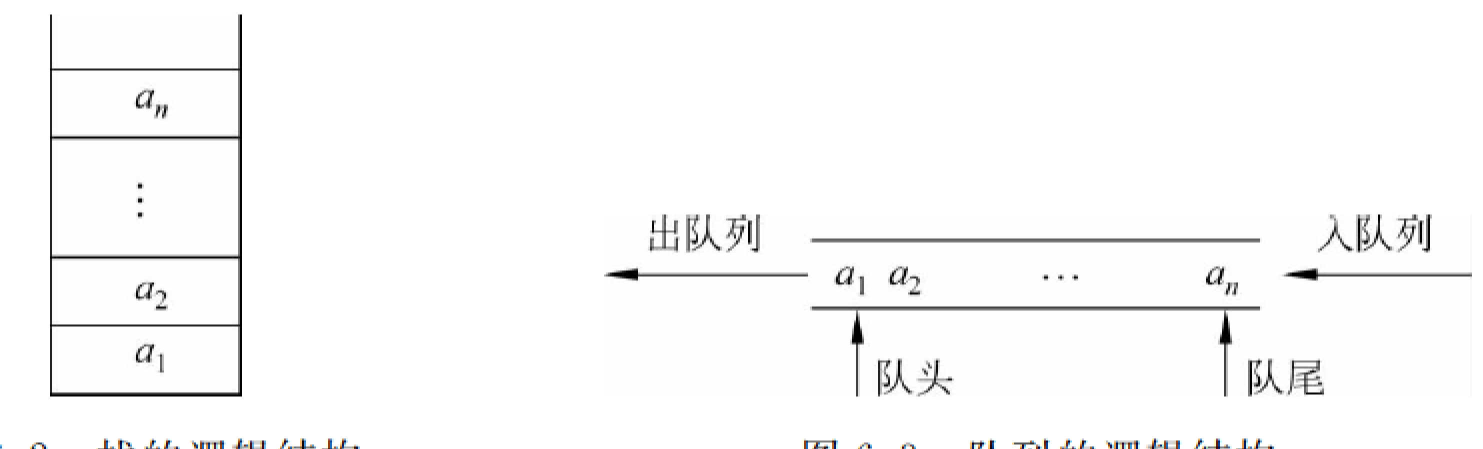


图 6.2　栈的逻辑结构　　图 6.3　队列的逻辑结构

2. 非线性结构

非线性结构是至少存在一个数据元素有两个或两个以上的直接后继(或直接前驱)的数据结构。线性结构一般有树和图。

1）树

在树形结构中,数据元素之间有着明显的层次关系,虽然每一层上的数据元素可能和下一层中多个元素(孩子) 相关,但只能和上一层中一个元素(双亲)相关。例如：人类的族谱。树的逻辑结构如图 6.4 所示。

二叉树的每个结点至多有两棵子树(不存在度大于 2 的结点),二叉树的子树有左右之分,次序不能颠倒,如图 6.5 所示。

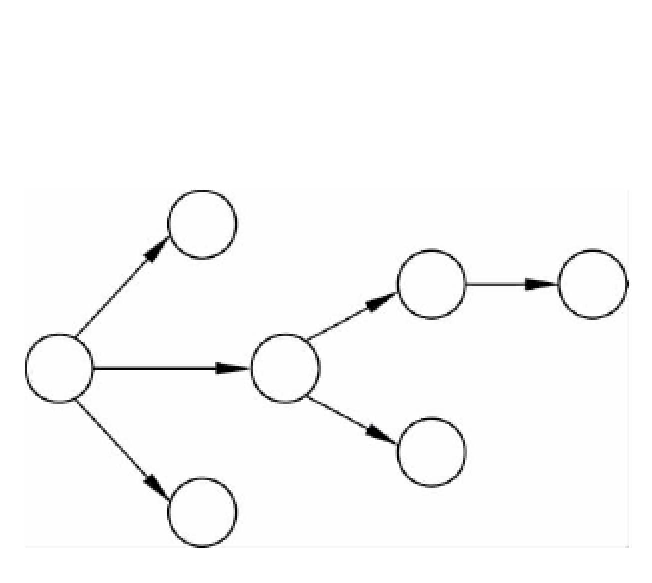


图 6.4　树的逻辑结构

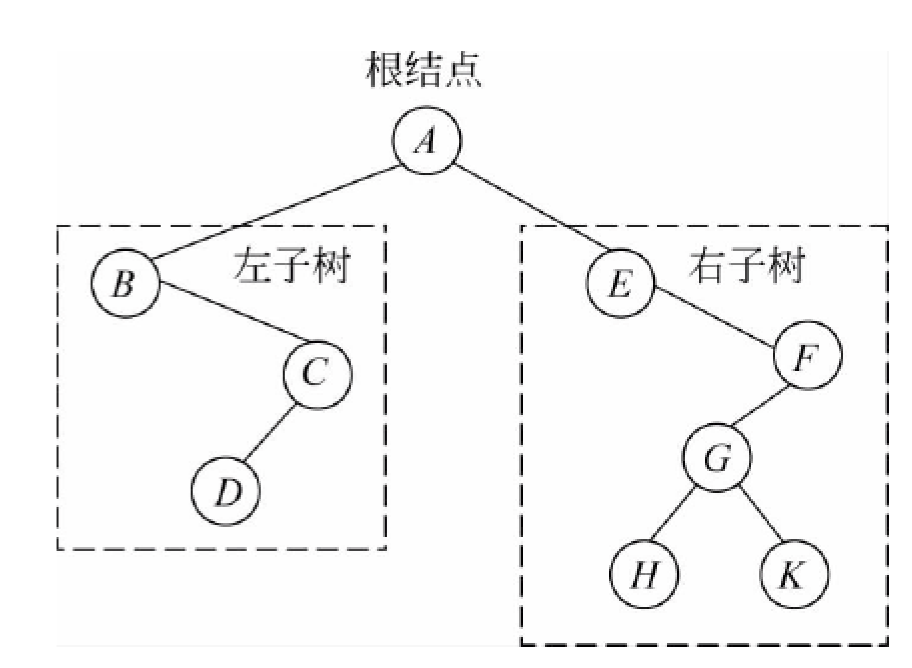


图 6.5　二叉树的逻辑结构

2）图

图在各个领域都有着广泛的应用,如交通路线等。在图结构中,任意两个数据元素之间都可能相关,因此可用于描述多对多的关系。图 G 由两个集合 V 和 E 组成,记为

$$G = (V, E)$$

其中：

- V 是顶点的有穷非空集合。
- E 是 V 中顶点偶对(称为边)的有穷集。

通常,也将图 G 的顶点集和边集分别记为 $V(G)$和 $E(G)$。$E(G)$可以是空集。若 E

(G)为空，则图 G 只有顶点而没有边。图的逻辑结构如图 6.6 所示。

在图 6.6 中：

- 顶点集 $V(G)=\{A, B, C, D, E\}$。
- 边集 $E(G)=\{$<A,B>, <A,E>, <B,C>, <C,D>, <D,A>, <D,B>, <E,C>$\}$。

图 6.6 图的逻辑结构

6.2.2 查找和排序

1. 查找

根据给定的某个值，在查找表中确定一个其关键字等于给定值的数据元素(记录)。若查找表中存在这样一个记录，则称“查找成功”，给出该记录在查找表中的位置；否则称“查找不成功”，查找结果给出“空记录”。

基于线性表的查找分为顺序查找、折半查找和分块查找等。

1) 顺序查找

顺序查找又称线性查找，是最基本的查找方法之一。其查找方法为：从表的一端开始，向另一端逐个用给定值 k 与关键码进行比较，若找到，查找成功，并给出数据元素在表中的位置；若整个表检测完，仍未找到与 k 相同的关键码，则查找失败，给出失败信息。

2) 折半查找

顺序查找的查找算法简单，但平均查找长度较大，特别不适用于较大的查找表。若以有序表表示查找表，则查找过程可以基于“折半”进行。折半查找的前提是线性表$(a_0, a_1, \cdots, a_{n-1})$已经按照从小到大的顺序排列。设要查找元素的关键字为 key，首先将查找范围的下限设为 low=0，上限为 high=$n-1$，其中点为 $m=\lfloor(\text{low}+\text{high})/2\rfloor$，中点元素记为 a_m。用 key 与中点元素 a_m 比较。若 key=a_m，该元素正为要找的元素，查找停止；若 key>a_m，则替换下限 low=(mid+1)，到下半段继续查找；若 key<a_m，则替换上限 high=mid−1，到上半段继续查找。依此循环，直至找到元素或 low>high 为止，low>high 时说明此元素未找到。

【例 6-1】 序列(1，2，5，7，8，11，14，20)的折半查找过程如图 6.7 所示，其中图 6.7(a)是查找关键字 14 成功的情况，图 6.7(b)是未找到关键字 15 的情况。

3) 分块查找

分块查找是顺序查找和折半查找的结合，性能介于顺序查找和折半查找之间，但无须像折半查找那样要求表中数据有序。设将数据$(a_0, a_1, \cdots, a_{n-1})$均分为 B 块，则前 $B-1$ 块中结点个数为 $s=\lfloor n/b\rfloor$，第 B 块的结点数为 $n-(B-1)\times s$。每一块中的数据无需有序，但要求“分块有序”，即前一块中的最大数据必须小于后一块中的最小数据。为此，构造一个索引表 index[1..B]，每个元素 index[i]($0\leqslant i\leqslant B-1$)中存放第 i 块的最大关键字 key、该块的起始位置 start 及结束位置 end，如图 6.8 所示。由索引确定记录所在的块，在块内进行查找。可见，分块查找的过程是一个逐步缩小搜索空间的过程。

2. 排序

排序是将一组无序的序列调整为有序的序列。例如，将关键字序列 (52，49，80，

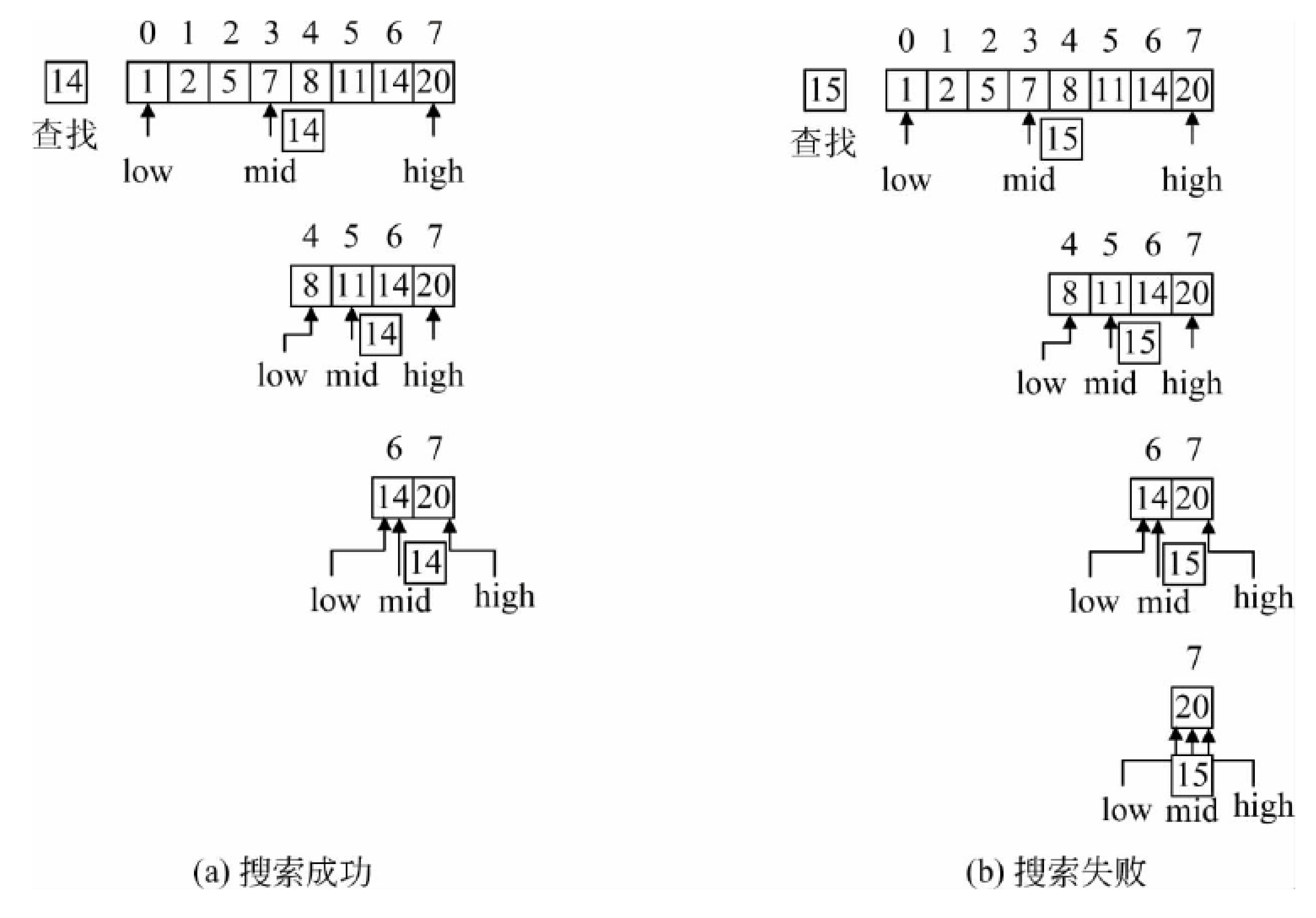


图 6.7　折半查找

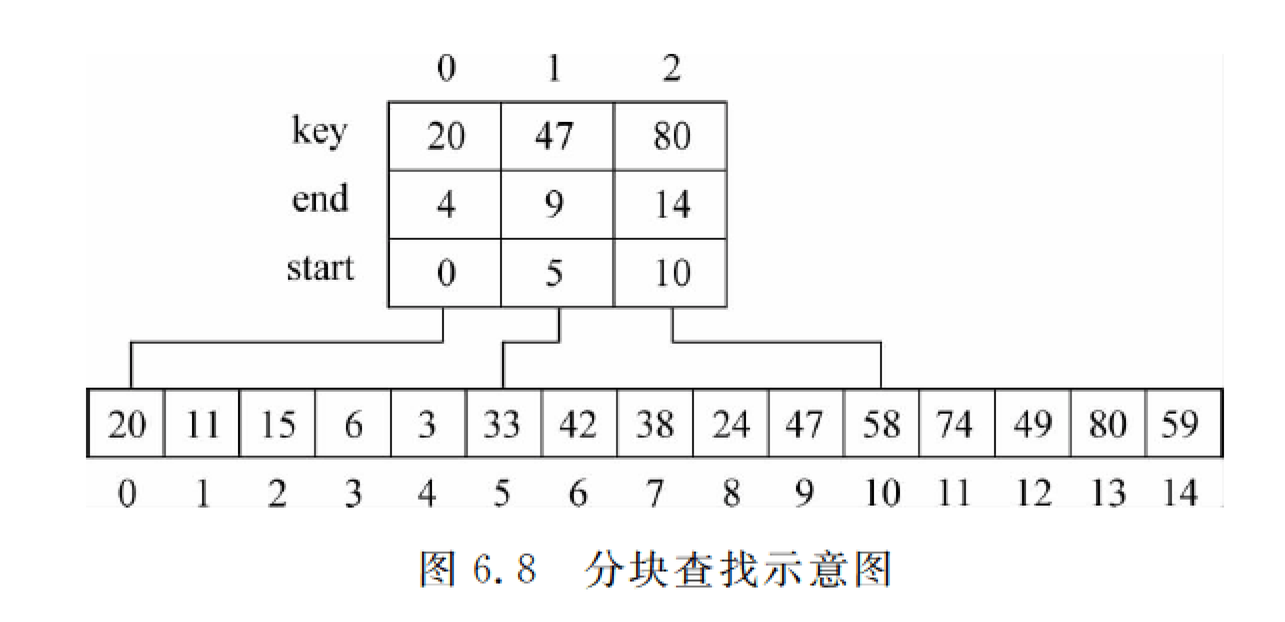


图 6.8　分块查找示意图

36，14，58，61，23，97，75)调整为(14，23，36，49，52，58，61，75，80，97)。排序的过程是一个逐步扩大有序子序列长度的过程，如图 6.9 所示。

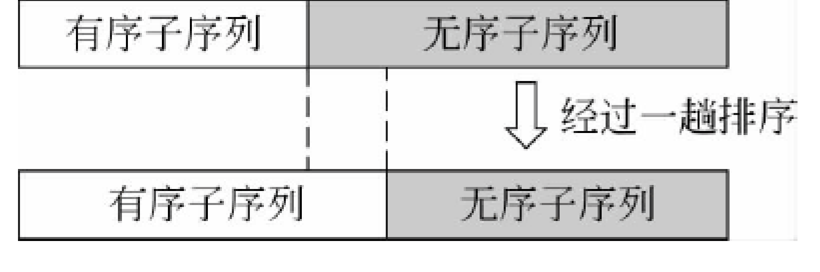


图 6.9　排序示意图

基于不同的扩大有序子序列长度的方法，内部排序方法大致可分下列几种类型：插入类、交换类、选择类和归并类。

1）插入类

将无序子序列中的一个记录插入到有序子序列中，增加有序子序列的长度。

- 基于顺序查找的插入类排序称为直接插入排序。
- 基于折半查找的插入类排序称为折半插入排序。
- 基于逐趟缩小增量的插入类排序称为希尔排序。

2）交换类

交换类排序通过交换无序子序列中的元素从而得到其中关键字最小或最大的元素，并将它加入有序子序列中，以此方法增加有序子序列的长度。

交换类排序分为冒泡排序、快速排序等。其中,冒泡排序法的过程如下:将待排序的数组元素看成是竖着排列的"气泡",最小的元素为最小的"气泡",较小的元素为较小的"气泡"。每一遍处理,就是自底向上检查一遍"气泡"序列,并时刻注意两个相邻元素的顺序是否正确。如果发现两个相邻"气泡"的顺序不对,例如轻的"气泡"在重的"气泡"的下面,则交换它们的位置。一遍处理之后,"最轻"的"气泡"就浮到了最高位置。第二遍处理之后,"次轻"的"气泡"就浮到了次高位置。如此反复,共处理 $n-1$ 次,就可以完成"气泡"的有序排列。

3)选择类

选择类排序从无序子序列中选择关键字最小或最大的元素,并将它加入到有序子序列中,以此方法增加有序子序列的长度。

选择类排序分为简单选择排序、树形选择排序和堆排序。其中,简单选择排序的基本思想是:给定一组无序数据 $a[0]$,$a[1]$,$a[2]$,…,$a[n-1]$,第一次从 $a[0]$~$a[n-1]$中选取最小值与 $a[0]$交换,第二次从 $a[1]$~$a[n-1]$中选取最小值与 $a[1]$交换……第 i 次从 $a[i-1]$~$a[n-1]$中选取最小值与 $a[i-1]$交换……第 $n-1$ 次从 $a[n-2]$~$a[n-1]$中选取最小值与 $a[n-2]$交换,总共经过 $n-1$ 次交换,得到一个从小到大排列的有序序列。

4)归并类

归并类排序通过归并两个或两个以上的有序子序列,逐步增加有序子序列的长度。

6.3 课后习题答案

1. 任意给一个 4 位数(各位数不完全相同),各个位上的数可组成一个最大数和一个最小数,这两个数的差又能组成一个最大数和一个最小数,直到某一步得到的差出现重复。写一个程序统计所有满足以上条件的 4 位数。例如:

3100 — 0013=3087

8730 — 0378=8352

8532 — 2358=6174

7641 — 1467=6174

【解答】

```
import random
num=random.randint(1000,9999)
cha=0
while cha!=6174:
qian=num//1000
bai=num//100-qian * 10
shi=num//10-qian * 100-bai * 10
ge=num %10
if qian>bai:
    t=qian;qian=bai;bai=t
if qian>shi:
```

```
        t=qian;qian=shi;shi=t
    if qian>ge:
        t=qian;qian=ge;ge=t
    if bai>shi:
        t=bai;bai=shi;shi=t
    if bai>ge:
        t=bai;bai=ge;ge=t
    if shi>ge:
        t=shi;shi=ge;ge=t
    min=qian*1000+bai*100+shi*10+ge
    max=ge*1000+shi*100+bai*10+qian
    cha=max-min
    print ("%d-%d=%d"%(max,min,cha))
    num=cha
```

运行结果：

```
6652-2566=4086
8640-468=8172
8721-1278=7443
7443-3447=3996
9963-3699=6264
6642-2466=4176
7641-1467=6174
```

2. $a+b$ 的 n 次幂的展开式中各项的系数很有规律，$n=2,3,4$ 时展开式中各项的系数分别是：1 2 1，1 3 3 1，1 4 6 4 1。这些系数构成了著名的杨辉三角形：

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5   10  10  5   1
```

编写代码，输入 n 值，输出杨辉三角形。

【解答】

```
def gettriangle(num):
    triangle=[]
    for i in range(0,num+1):
        tmp=[1]*(i+1)
        for j in range(1,i):
            tmp[j]=triangle[i-1][j-1]+triangle[i-1][j]
        triangle.append(tmp)
    return triangle
def printtriangle(triangle,width=4):
    column=len(triangle[-1])*width;
```

```
    for sublist in triangle:
        result=[]
        for element in sublist:
            tmp='{0:^{1}d}'.format(element,width)
            result.append(tmp)
        print('{0:^{1}s}'.format(''.join(result),column))

if __name__=='__main__':
    num=int(input("Enter a nonnegative integer:"))
    triangle=gettriangle(num)
    n=len(str(triangle[-1][len(triangle[-1])//2]))
    printtriangle(triangle,3+n)
```

运行结果：

```
Enter a nonnegative integer:5
            1
         1     1
      1     2     1
   1     3     3     1
1     4     6     4     1
1     5    10    10     5     1
```

3. 如果有两个数，其中任一个数除它本身以外的所有因数的和正好等于另一个数，则称这两个数为互满数。求出3000以内所有的互满数并输出。

【解答】

```
def appsum(x):
    s=0;i=1
    while i<x:
        if(x%i==0):
            s+=i
        i+=1
    return s
for i in range(1,3000):
    j=appsum(i)
    if(i>j and appsum(j)==i):
        print "%d"%i
        print "%d\n"%j
```

运行结果：

```
284 220
1210 1184
2924 2620
```

4. 验证任意一个大于 5 的奇数可表示为 3 个素数之和。

【解答】

```
def prime(n):
    i=2
    while i<n:
        if(not n%i ):
            break
        i+=1
    return n>1 and i>=n
n=input("please input a number:")
for i in range(2,n-4+1):
    for j in range(i,n-i*2+1):
        for k in range(j,n-i-j+1):
            if(i+j+k==n and prime(i) and prime(j) and prime(k)):
                print("%d=%d+%d+%d\n" % (n,i,j,k))
```

运行结果：

```
please input a number: 19
19=3+3+13
19=3+5+11
19=5+7+7
```

6.4　习题与解答

6.4.1　习题

1. 若将某素数的各位数字顺序颠倒后得到的数仍是素数，则此数称为可逆素数。求出 100 以内的可逆素数。

2. 将一个正整数分解质因数。例如：输入 90，打印出 90＝2＊3＊3＊5。

6.4.2　习题参考答案

1. 若将某素数的各位数字顺序颠倒后得到的数仍是素数，则此数为可逆素数。求出 100 以内的可逆素数。

【解答】

```
def isprime(num):
    flag=1;i=2
    while i < num:
        if num %i==0:
            flag=0
            break
        i=i+1
```

```
    if flag==1:
        return True,
def rev(n):
    r=0
    while (n>0):
        r=r * 10+(n %10)
        n //=10
    return r

for i in range(3,100):
    if(isprime(i)):
        if isprime(rev(i)):
            print("%d" %i)
```

运行结果：

```
3 5 7 11 13 17 31 37 71 73 79 97
```

2. 将一个正整数分解质因数。例如，输入 90，输出 90＝2＊3＊3＊5。

【解答】

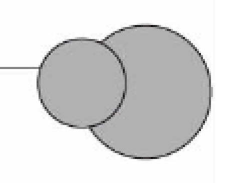
```
def main():
    n=int(raw_input('Enter a number:'))
    print n,'='
    while(n!=1):
        for i in range(2,n+1):
            if (n%i)==0:
                n/=i
                if(n==1):
                    print '%d'%(i)
                else:
                    print '%d * '%(i)
                break

if __name__=="__main__":
    main()
```

运行结果：

```
Enter a number: 30
30=2* 3* 5
```

第7章 函数与模块



7.1 本章要求

- 掌握函数的声明和调用。
- 掌握实参与形参。
- 掌握变量的作用域。
- 理解递归的特点。
- 了解参数类型及其模块。

7.2 本章知识重点

7.2.1 函数的概念

Python 的函数分为系统函数和用户自定义函数。系统函数又称内置函数或内建函数。用户自己创建函数称为用户自定义函数。一般来说，函数的大小为 70～200 行代码。

函数具有以下优点：

（1）简化程序的结构，提高程序的可读性。

（2）函数代码可以重复调用，减少了程序中大量重复代码的书写，函数一旦创建，便可以在程序的任何一个地方调用。

（3）使得应用程序更容易调试、修改和维护。

（4）便于多人协同合作开发。

7.2.2 函数声明和调用

在 Python 中，函数声明语法格式如下：

```
def <函数名> ([<形参列表>]):
    [<函数体>]
```

说明：

（1）函数使用关键字 def(define 的缩写）声明，函数名为有效的标识符，形参列表为函数的参数。

（2）函数没有明显的 begin 和 end，没有标明函数的开始和结束的花括号。唯一的分隔符是一个冒号（:）。

（3）函数名下的每条语句前都要用 Tab 键缩进，没有缩进的第一行则被视为在函数体之外的语句，与函数同级的程序语句。

（4）函数没有定义返回的数据类型，函数通过 return 语句返回指定的值，否则将返回空值（none）。

【例 7-1】 利用海伦公式求三角形面积。

```
import math
def triarea(x,y,z):
    s= (x + y + z) / 2
    print math.sqrt((s- x) * (s- y) * (s- z) * s)
#主函数
triarea(3,4,5)
```

【解析】

triarea(3,4,5)调用 triarea(x,y,z)，程序执行步骤如图 7.1 所示。

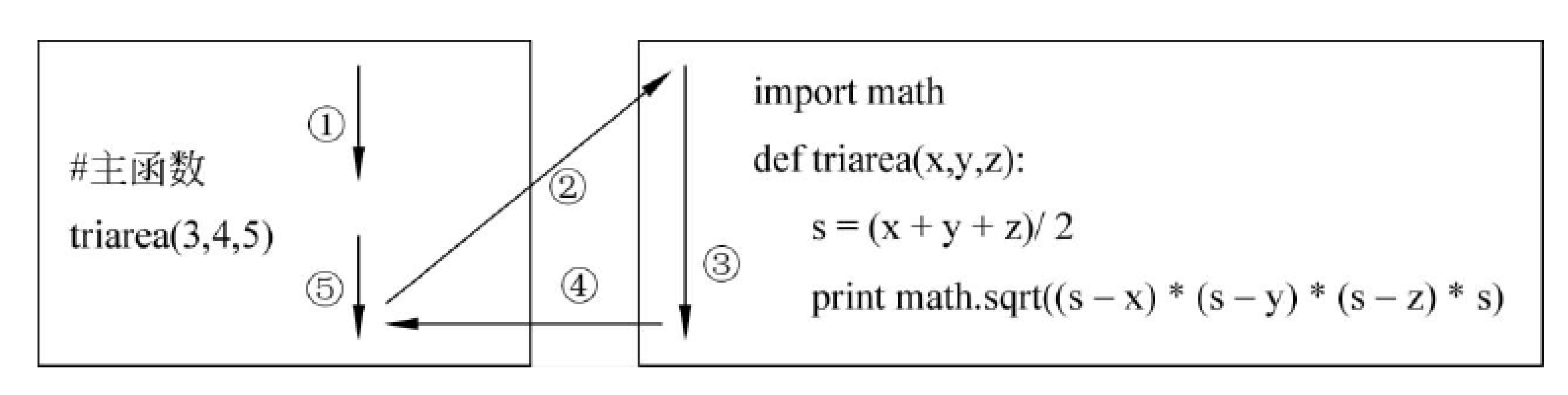


图 7.1 函数调用

7.2.3 函数的参数

Python 函数有必备参数、命名参数、默认参数和可变长参数 4 种参数类型。

1. 必备参数

必备参数须以正确的顺序传入函数。调用时的数量必须和声明时的一样。

【例 7-2】 必备参数举例。

```
def printme(str):
    print str;
    return;
```

调用该函数：

```
printme();
```

运行结果：

```
Traceback (most recent call last):
  File "test.py", line 8, in <module>
```

```
    printme();
TypeError: printme() takes exactly 1 argument (0 given)
```

【解析】 调用 printme()函数，必须传入一个参数，否则会出现语法错误。

2. 关键参数

关键参数又称为命名参数，Python 解释器能够用参数名匹配参数值，调用方根据参数的命名确定传入的参数值，可以跳过不传的参数或者乱序传参。

【例 7-3】 命名参数举例。

```
def printinfo(name, age):
    print "Name: ", name;
    print "Age ", age;
    return;
```

调用该函数：

```
printinfo(age=24, name="zhou");
```

运行结果：

```
Name: zhou
Age  24
```

3. 默认参数

默认参数又称为缺省参数。调用函数时，参数的值如果没有传入，则被认为是默认值。

【例 7-4】 默认参数举例。

```
def printinfo(name, age=20):
    print "Name: ", name;
    print "Age ", age;
    return;
```

调用该函数：

```
printinfo(age=24, name="zhou");
printinfo(name="zhou");
```

运行结果：

```
Name: zhou
Age  24
Name: zhou
Age  20
```

4. 可变长参数

可变长参数又称不定长参数。参数以 * 开头代表一个任意长度的元组，可以接收连续一串参数。参数以**开头代表一个字典，参数的形式是 key=value，接受连续的任意多个参数。

【例 7-5】 不定长参数举例。

```
def printinfo(arg1, *vartuple):
    print "print:"
    print arg1
    for var in vartuple:
        print var
    return;

printinfo(10);
printinfo(20, 30, 40);
```

运行结果：

```
print:
10
print:
20
30
40
```

7.2.4 递归函数

通过递推关系将原来的问题缩小成一个规模更小的同类问题，并延续这一缩小规模的过程，直到在某一规模上问题的解是已知的；然后开始返回，最终解决问题，这种思想称为递归。

递归函数就是在自定义函数内部调用其自己，包含递推和回归两个过程。构成递归的条件如下：

（1）有确定的递归结束条件和结束时的值。

（2）能用递归形式表示，并且递归向结束条件发展。

【例 7-6】 计算 4 的阶乘。

4 的阶乘定义为 4 乘以 3 的阶乘，即 4!=4×3!；3 的阶乘定义为 3 乘以 2 的阶乘，即 3!=3×2!。依此类推。

```
def fact(n):
    if n==1:
        return 1
    return n*fact(n-1)
```

主函数如下：

```
Print fact(4)
```

fact(4)递归求解的过程如图 7.2 所示。

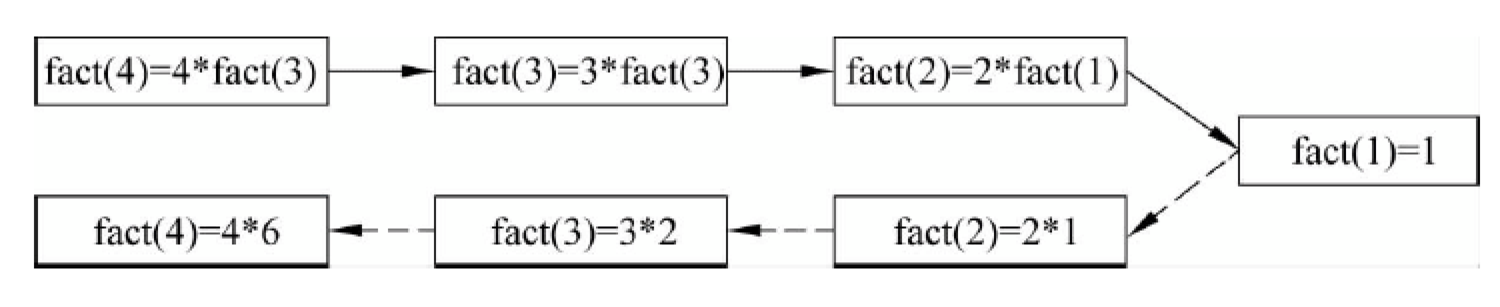


图 7.2　fact(n)的递归求解过程

7.2.5　变量作用域

变量作用域是指变量有效的范围，Python 具有局部变量和全局变量。局部变量是指定义在函数体内的变量，只能在本函数内部访问。全局变量又称公用变量，使用关键字 global 声明，可以在整个程序范围内访问。

【例 7-7】　变量作用域举例。

```
total= 0; # This is global variable.
def sum(arg1, arg2):
    total= arg1 + arg2; # total is local variable
    print "Inside the function local total: ", total
    return total;
sum(10, 20);
print "Outside the function global total: ", total
```

运行结果：

```
Inside the function local total : 30
Outside the function global total : 0
```

7.3　课后习题答案

1. 编制判断素数的 Sub 函数或 Function 函数，验证哥德巴赫猜想：一个不小于 6 的偶数可以表示为两个素数之和。例如，6＝3＋3，8＝3＋5，10＝3＋7。

【解答】

```
def IsPrime(val):
    i= 2
    while i< = val:
        if val %i= = 0:
            break
        i+ = 1
    if i= = val:
```

```
        return True
    else:
        return False
b=input("Please input a even")
if b %2==0:
    i=1
    while i<=b:
        j=b-i
        if (IsPrime(j)):
            if (IsPrime(b-j)):
                print "%d + %d=%d\n" %(j, b-j, b)
        i+=1
else:
    print "no even"
```

运行结果：

```
Please input a even8
5+3=8
3+5=8
```

2. 实现求两数中较大数的函数。

【解答】

```
def max(a,b):
    if a>b:
        return a
    else:
        return b
a=input("please input a number")
b=input("please input a number")
print max(a,b)
```

运行结果：

```
please input a number4
please input a number8
8
```

3. 实现计算表达式 $1+3+\cdots+(2n-1)$ 值的函数。

【解答】

```
def sum(n):
    i=1;s=0
    while i<=2*n-1:
        s=s+i
        i=i+2
```

```
    print s
n= input("please input a number")
sum(n)
```

运行结果：

```
please input a number5
25
```

4. 完成一个函数，将所给的(1，2，3，－5，－4，5，9，－8，－1)重新排列，使得所有负数都在正数的左边。

【解答】

```
def answer():
    num= [1, 2, 3,- 5,- 4, 5, 9,- 8,- 1]
    num.sort()
    print num;
answer()
```

运行结果：

```
[- 8,- 5,- 4,- 1, 1, 2, 3, 5, 9]
```

7.4　习题与解答

7.4.1　习题

1. 假设有 4 种币值，面值分别为二角五分、一角、五分和一分。现在要找给某顾客六角三分钱。怎样找零钱才能给顾客的硬币个数最少？

2. 创建函数 swap(a,b)，功能如下：比较 a 和 b 两个数的大小，若 a 小于 b，则交换两个数的位置。

3. 输入一个年份，判断是否为闰年。

4. 利用递归函数，将输入的 5 个字符，以相反顺序打印出来。

7.4.2　习题参考答案

1. 假设有 4 种币值，面值分别为二角五分、一角、五分和一分。现在要找给某顾客六角三分钱。怎样找零钱才能给顾客的硬币个数最少？

【解析】　找零钱的基本思路是：每次都选择面值不超过需要找给顾客的钱的最大面值的硬币。

(1) 选取不超过六角三分钱的最大面值硬币二角五分，剩三角八分。

(2) 选取不超过三角八分的最大面值硬币二角五分，剩一角三分。

(3) 选取不超过一角三分的最大面值硬币一角，剩三分。

(4) 选取不超过三分的最大面值硬币一分，剩二分。

(5) 选取不超过二分的最大面值硬币一分，剩一分。

(6) 找一分，结束。

因此，找零钱的结果是 2 个二角五分、1 个一角、3 个一分，共需 6 枚硬币。

代码如下：

```
v=[25,10,5,1]
n=[0,0,0,0]
def change():
    T_str=input("please input change")
    T=int(T_str)
    greedy(T)
    for i in range(len(v)): print("give",v[i],"change",n[i])
    s=0
    for i in n:s=s+i
    print('the least change',s)
def greedy(T):
    if T==0:return
    elif T>v[0]:
        T=T-v[0];n[0]=n[0]+1
        greedy(T)
    elif v[0]>T>v[1]:
        T=T-v[1];n[1]=n[1]+1
        greedy(T)
    elif v[1]>T>v[2]:
        T=T-v[2];n[2]=n[2]+1
        greedy(T)
    else:
        T=T-v[3];n[3]=n[3]+1
        greedy(T)
if __name__=="__main__":
    change()
```

运行结果如图 7.3 所示。

```
C:\WINDOWS\system32\cmd.exe - PAUSE
please input change63
('give', 25, 'change', 2)
('give', 10, 'change', 1)
('give', 5, 'change', 0)
('give', 1, 'change', 3)
('the least change', 6)
请按任意键继续. . .
```

图 7.3　题 1 程序运行结果

2. 创建函数 swap(a,b),功能如下:比较 a 和 b 两个数的大小,若 a 小于 b,则交换两个数的位置。

【解答】

```
x,y=input('please input two number: ')
print "before swap:"
print x,y
def swap(a,b):
    if a<b:
        a,b=b,a
        print "after swap:"
        print a,b
swap(x,y)
```

运行结果:

```
please input two number: 2,5
before swap:
2 5
after swap:
5 2
```

3. 输入一个年份,判断是否为闰年。

【解答】

```
def leapyear(a):
    if a%400==0 or (a%4==0 and a%100!=0):
        print '%d is a leap year!'%a
    else:
        print '%d is not a leap year!'%a
year=input('please input a year number:')
leapyear(year)
```

运行结果:

```
please input a year number:2016
2016 is a leap year!
```

4. 利用递归函数,将输入的 5 个字符以相反顺序打印出来。

【解答】

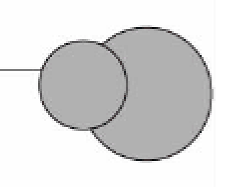
```
def pai(n):
    next=0
    if n<=1:
        next=input("please input a number")
        print next
    else:
```

```
        next= input("please input a number")
        pai(n- 1)
        print next
i= 5
pai(i)
```

运行结果：

```
please input a number1
please input a number2
please input a number3
please input a number4
please input a number5
5 4 3 2 1
```

第8章 面向对象程序设计基础



8.1 本章要求

- 了解面向对象的基本概念。
- 掌握类的声明、对象的创建和使用。
- 掌握类属性与实例属性。
- 掌握构造函数与析构函数。
- 掌握类的继承。
- 了解多态性。

8.2 本章知识重点

8.2.1 对象三大特性

对象有封装性、继承性和多态性三大特性，下面依次进行介绍。

1. 封装性

类是对客观事物的抽象，是一组具有相同属性和操作的对象的集合。封装具有对内部细节隐藏保护的能力，保证了类具有较好的独立性，防止外部程序破坏类的内部数据，同时便于程序的维护和修改。

封装性使得对象具有抽象性。抽象性是指将具有一致的数据结构（属性）和行为（操作）的对象抽象成类，反映与应用有关的重要性质，而忽略其他无关内容。

2. 继承性

继承是一种连接类与类的层次模型，是利用现有类派生出新类的过程。新类拥有原有类的特性，又增加了自身新的特性。继承性简化了类和对象的创建工作，增强了代码的可重用性。

3. 多态性

多态性是指同一种事物具有多种形态。例如，某个属于“形状”基类的对象，在调用它

的计算面积方法时，程序会自动判断出它的具体类型。如果是圆，则调用圆对应的计算面积方法；如果是正方形，则调用正方形对应的计算面积方法。

多态性允许每个对象以适合自身的方式响应共同的消息，不必为相同功能的操作作用于不同的对象而特意识别，为软件开发和维护提供了极大的方便。

8.2.2 类与对象

Python 使用 class 关键字构造类，并在类中定义属性和方法。通常认为类是对象的模板，对象是类创建的产品，对象是类的实例。

声明类的语法格式如下：

```
class 类名:
属性定义 #变量定义
方法定义 #函数定义
```

说明：

(1) 定义类的关键字为 class，类名通常第一个字母大写。

(2) 对象通过类名后跟一对圆括号来创建。

(3) 类具有属性和方法。

8.2.3 继承性

在面向对象(OO)程序设计中，继承性是通过派生类和基类实现的，被继承的基类称为父类或超类(base class、super class)，而新类称为子类或派生类(subclass)。

Python 中实现继承的语法如下：

```
class SubClassName(ParentClass1[, ParentClass2, …]):
    class_suite
```

说明：

(1) 基类只是简单地列在类名后面的小括号里。

(2) Python 支持多重继承，即派生类继承多个基类，只需要在类名后面的小括号中列出多个类名，以逗号分隔即可。

在 Python 中实现继承有如下规则：

(1) 基类的构造(__init__方法)不会被自动调用，必须在派生类中显式调用父类的__init__方法。

(2) 调用基类的方法，需要加上基类的类名作为前缀，带上 self 参数变量，而在类中调用普通函数时并不需要带上 self 参数。

8.3 课后习题答案

1. 如何实现类的继承？

【解答】 学校中的教师和学生有一些共同属性，比如姓名、年龄和地址。教师和学生

也有自身专有的属性,例如教师的薪水、课程和假期,学生的成绩。如果将教师和学生创建为两个独立的类,要增加一个新的共有属性,就意味着要在这两个独立的类中都增加。较好的方法是创建一个共同的类,称为 SchoolMember,教师和学生的类继承这个共同的类。当要为教师和学生都增加一个新的身份证域时,只需将其增加到 SchoolMember 类中即可。在 SchoolMember 类中的任何改变都会自动地反映到子类中。

关于实现继承性的语法及规则可参考 8.2.3 节的内容。

2. 如何在类中定义属性?

【解答】 类属性是在类中的方法之外定义的属性,又分为公有属性和私有属性,不像 C++ 通过 public 和 private 关键字区别公有属性和私有属性,Python 是以属性命名方式来区分的,如果在属性名前面加了两个下划线“__”,则表明该属性是私有属性,否则为公有属性。

3. 如何实现方法的重载?

【解答】 方法重载实际上就是在子类中使用方法名与父类的方法名完全相同,从而重载父类的方法。如果仍需使用父类的该方法,需要在父类所生成的对象加“.”和方法名的形式调用。

4. 编写人员类(Person),该类具有姓名(Name)、年龄(Age)、性别(Sex)等域。然后通过对 Person 类的继承得到一个教师类(Teacher),该类能够存放教师的职称、学历、工资、奖金等信息,并能计算出总收入(工资+奖金),要求对该类构造函数进行重载。

【解答】

```
class Person:
    def __init__(self,Name,Age,Sex):
        self.Name= Name
        self.Age= Age
        self.Sex= Sex
class Teacher(Person):
    sum= 0
    def __init__(self,Name,Age,Sex,Title,Qual,Salary,Prize):
        self.Name= Name
        self.Age= Age
        self.Sex= Sex
        self.Title= Title
        self.Qual= Qual
        self.salary= Salary
        self.Prize= Prize
    def p(self):
        print "name:",self.Name
        print "age:", self.Age
        print "sex:",self.Sex
        print "Title:",self.Title
        print "Qual:",self.Qual
```

```
        print "Salary:",self.salary
        print "Prize:",self.Prize
    def count(self):
        sum= self.salary+ self.Prize
        print "sum is",sum
tea= Teacher('li ming', 20,'male',"professor","doctor",5500,2000)
tea.p()
tea.count()
```

运行结果：

```
name: li ming
age: 20
sex: male
Title: professor
Qual: doctor
Salary: 5500
Prize: 2000
sum is 7500
```

8.4 习题与解答

8.4.1 习题

1. 父类 SchoolMember 具有姓名、年龄属性，子类 Teacher 具有薪水属性，子类 Student 具有成绩属性。请编程实现。

2. 实现如下功能：学校成员类具有成员的姓名和总人数属性。老师类继承学校成员类，具有工资属性。学生类继承学校成员类，具有成绩属性。创建对象时，总人数增一；对象注销时，总人数减一。

8.4.2 习题参考答案

1. 父类 SchoolMember 具有姓名、年龄属性，子类 Teacher 具有薪水属性，子类 Student 具有成绩属性。请编程实现。

【解答】

```
class SchoolMember:
    def __init__(self,name,age):
        self.name= name
        self.age= age
        print 'init SchoolMember: ', self.name
    def tell(self):
        print 'name:%s; age:%s' % (self.name, self.age)
class Teacher(SchoolMember):
    def __init__(self,name,age,salary):
```

```
        SchoolMember.__init__(self,name,age)
        self.salary= salary
        print 'init Teacher: ', self.name
    def tell(self):
        SchoolMember.tell(self)
        print 'salary: ', self.salary
class Student(SchoolMember):
    def __init__(self,name,age,marks):
        SchoolMember.__init__(self,name,age)
        self.marks= marks
        print 'init Student: ', self.name
    def tell(self):
        SchoolMember.tell(self)
        print 'marks: ', self.marks
t= Teacher('yanggang', 20, 1000)
s= Student('liming', 12, 86)
members= [t,s]
print
for member in members:
    member.tell()
```

运行结果：

```
init SchoolMember: yanggang
init Teacher: yanggang
init SchoolMember: liming
init Student: liming

name:yanggang; age:20
salary: 1000
name:liming; age:12
marks: 86
```

2. 实现如下功能：学校成员类具有成员的姓名和总人数属性。老师类继承学校成员类，具有工资属性。学生类继承学校成员类，具有成绩属性。创建对象时，总人数增一；对象注销时，总人数减一。

```
#!/usr/bin/python
#-*-coding: utf-8-*-
class SchoolMember:
    sum_member= 0
    def __init__(self, name):
        self.name= name
        SchoolMember.sum_member += 1
        print "name %s" %self.name
        print "count %d" %SchoolMember.sum_member
```

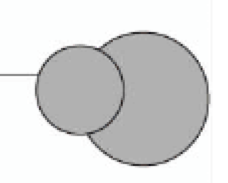
```
    def say_hello(self):
        print "My name is:%s" %self.name
    def __del__(self):
        SchoolMember.sum_member-=1
        print "%s left, count is %d" %(self.name, SchoolMember.sum_member)
class Teacher(SchoolMember):
    def __init__(self, name, salary):
        SchoolMember.__init__(self, name)
        self.salary=salary
    def say_hello(self):
        SchoolMember.say_hello(self)
        print "I am a teacher,salary: %d" %self.salary
    def __del__(self):
        SchoolMember.__del__(self)
class Student(SchoolMember):
    def __init__(self, name, mark):
        SchoolMember.__init__(self, name)
        self.mark=mark
    def say_hello(self):
        SchoolMember.say_hello(self)
        print "I am a student,Mark:%d" %self.mark
    def __del__(self):
        SchoolMember.__del__(self)
t=Teacher("lao zhou", 3000)
t.say_hello()
s=Student("xiao wang", 77)
s.say_hello()
```

运行结果：

```
name lao zhou
count 1
My name is:lao zhou
I am a teacher,salary: 3000
name xiao wang
count 2
My name is:xiao wang
I am a student,Mark:77
xiao wang left, count is 1
lao zhou left, count is 0
```

第 9 章

文　件



9.1　本章要求

- 了解文件的概念。
- 了解字符编码。
- 掌握文件的读取和写入。
- 了解存储器。

9.2　本章知识重点

9.2.1　字符编码

电子邮件中汉字往往显示为一串问号或者乱码，这是因为发信人和收信人使用的编码方式不一样。字符编码是计算机技术的基石，常见的字符编码有 ASCII、UTF-8、Unicode、GB2312、GBK 等。

1. ASCII 编码

在计算机内部，所有的信息最终都表示为一个二进制的字符串。每一个二进制位(bit)有 0 和 1 两种状态，8 个二进制位就可以组合出 256 种状态，称为一个字节(byte)。ASCII 编码将英语字符与二进制值之间的关系进行了规定，对 0～9 的 10 个数字、26 个大小写字英文字母及一些其他符号进行了编码。

2. GB2312 编码

汉字多达 10 万个左右，而 ASCII 编码只能表示 256 种符号，远远不够，因此简体中文使用 GB2312 编码方式，使用两个字节表示一个汉字。

3. Unicode 编码

Unicode 编码将世界上的每一个符号进行独一无二的编码，解决了乱码问题。Unicode 的全称是 Universal Multiple-Octet Coded Character Set，简称 Unicode Character Set(Unicode 字符集，UCS)。Unicode 又称为抽象编码，只是一个符号集，规定

了符号的二进制代码,并没有规定这个二进制代码应该如何存储和传输。传输编码是由UTF(UCS Transformation Format,Unicode字符集传输格式)规范规定,常见的UTF规范包括UTF-8、UTF-16。

4. UTF编码

浏览网页的源码上会有类似<meta charset="UTF-8"/>的信息,表示该网页为UTF-8编码。UTF-8作为互联网上使用最广的Unicode编码的实现方式之一,以8位(1B)表示英语,以24b(3B)表示中文及其他语言。

9.2.2 文件分类

Python语言根据文件编码方式不同将文件分为文本文件和二进制文件。

文本文件又称为ASCII文件,是由ASCII编码字符组成并且不带任何格式的文件,通常使用字处理软件(如Windows记事本等)编辑。文本文件的读取必须从文件的头部开始,一次全部读出,不能只读取中间的一部分数据,不可以跳跃式访问。文本文件的每一行文本相当于一条记录,每条记录可长可短,记录之间使用“换行符”进行分隔,不能同时进行读、写操作。文本文件的优点是使用方便,占用内存资源较少,但其访问速度较慢,并且不易维护。

二进制文件是最原始的文件类型,直接把二进制码存放在文件中,以字节为单位访问数据,不能用字处理软件进行编辑。二进制文件允许程序按所需的任何方式组织和访问数据,也允许对文件中各字节数据进行存取和访问。

除此之外,根据存储数据的性质可以将文件分为程序文件和数据文件,根据文件的流向分为输入文件和输出文件,根据文件的存储介质分为磁盘文件、磁带文件等。

9.2.3 文件读写操作

1. 文件的读取

Python提供了read()、readline()和readlines()方法用于文本文件的读取。

方式1:按行读取方式——readline()。

```
fd=open(filename, "r")
while True:
    line=fd.readline()
    if line:
        print line                  #保留换行符
    else:
        break
fd.close()
```

方式2:多行读取方式——readlines()。

```
fd=file(filename, "r")              #等价于open
```

```
lines= fd.readlines()
for line in lines:
    print line
fd.close()
```

方式 3：一次性读取方式——read()。

```
fd= open(filename, "r")
content= fd.read()                #读取文件所有内容
print content
fd.close()
fd= open(filename, "r")
print "length:", fd.tell()        #返回文件的当前位置读写指针
fd.seek(0)                        #文件指针返回文件开头,否则再读取不到内容
content= fd.read(5)               #读取前 5 个字节内容
print content,
fd.close()
```

2. 文件的写入

Python 提供了 write()和 writelines()函数将数据写入到文件中。

方式 1：write()把字符串写入文件。

```
fd= file(filename, "w+ ")         #以写入方式打开,清除文件原来的内容
fd.write("goodbye\n")
fd.close()
```

方式 2：writelines()把列表中存储的内容写入文件。

```
fd= file(filename, "a+ ")         #以追加方式打开
content= ["helloworld\n", "how are you\n"]
fd.writelines(content)            #以将列表内容写入文件中
fd.close()
```

9.3　课后习题答案

1. 什么是文件？文件分为哪几类？

【解答】 文件是指在各种存储介质上永久存储的数据集合，如 Word 的.doc 文件，将其保存在磁盘上就是磁盘文件，输出到打印机上就是打印机文件。

Python 语言根据文件编码方式不同将文件分为文本文件和二进制文件。

(1) 文本文件。又称为 ASCII 文件，是由 ASCII 编码字符所组成并且不带任何格式的文件，通常使用字处理软件(如 Windows 记事本等)编辑。文本文件的读取必须从文件的头部开始，一次全部读出，不能只读取中间的一部分数据，不可以跳跃式访问。文本文件的每一行文本相当于一条记录，每条记录可长可短，记录之间使用“换行符”进行分隔，

不能同时进行读、写操作。文本文件的优点是使用方便，占用内存资源较少，但其访问速度较慢，并且不易维护。

（2）二进制文件。是最原始的文件类型，直接把二进制码存放在文件中，以字节为单位访问数据，不能用字处理软件进行编辑。二进制文件允许程序按所需的任何方式组织和访问数据，也允许对文件中各字节数据进行存取和访问。

除此之外，根据存储数据的性质可以将文件分为程序文件和数据文件，根据文件的流向分为输入文件和输出文件，根据文件的存储介质分为磁盘文件、磁带文件等。

2. Python 提供了几种文件访问的方法？分别是什么？

【解答】 略。

3. 文本文件和二进制文件有什么异同点？

【解答】 略。

4. 文本文件的读写函数有哪些？分别解释其功能和参数。

【解答】 略。

5. 二进制文件的读写函数有哪些？分别解释其功能和参数。

【解答】 略。

6. 存储器是什么？如何进行文件的读写操作？

【解答】 Python 提供了 pickle 模块用于在一个文件中存储和读取数据。另外，还有一个 cPickle 模块，其功能和 pickle 模块完全相同，只不过 cPickle 是用 C 语言编写的，执行速度较快。一般统称两个模块为 pickle 模块，用于持久地存储对象。

pickle 模块存储对象的过程如下：首先以写模式打开一个 file 对象，然后调用 dump 函数把对象转换为字符串(bytes)存储到文件中，这个过程称为存储。pickle 模块的 load 函数用于从文件中取回对象，这个过程称为取存储。

9.4 习题与解答

9.4.1 习题

1. 从键盘输入字符等内容，保存到 txt 文件中。

2. 创建一个 txt 文件，在其中写入学生的基本信息，包括姓名、性别、电话和地址 4 个信息。

9.4.2 习题参考答案

1. 从键盘输入字符等内容，保存到 txt 文件中。

【解答】

```
str= raw_input("please input a string")
f= open('d:\poem.txt', 'w')              # open for 'w'riting
f.write(str)                             # write text to file
f.close()                                # close the file
```

程序运行结果如图 9.1 所示。

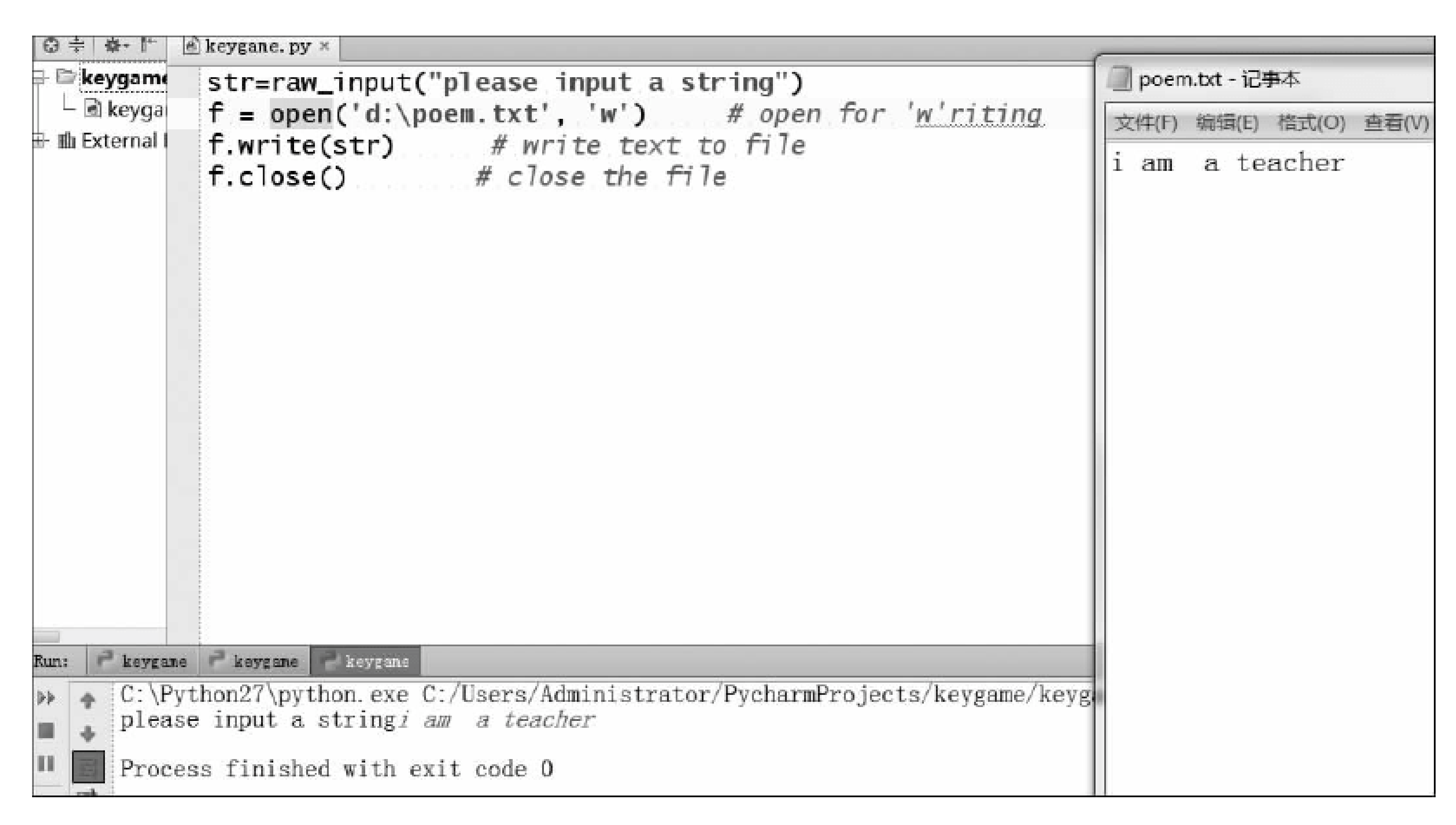


图 9.1　题 1 程序运行结果

2. 创建一个 txt 文件，在其中写入学生的基本信息，包括姓名、性别、电话和地址 4 个信息。

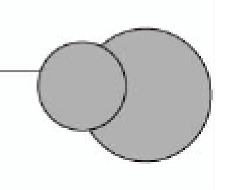
【解答】

```
__author__ = 'Administrator'
f= open('d:\\filecreat.txt','w')
f.write('name'+ '\tsex'+ '\tcellphone'+ '\t\taddress'+ '\n')
flag= 1
while flag= = 1:
    name= raw_input('please input a name:')
    sex= raw_input('please input the sex:')
    phone= raw_input('please input the phone number:')
    address= raw_input('please input the address:')
    s= name+ '\t'+ sex+ '\t'+ phone+ '\t\t'+ address+ '\n'
    f.write(s)
    flag= input('will you go on?(1/0):')
f.close()
```

运行结果：

```
please input a name:zhou
please input the sex:nan
please input the phone number:13319293232
please input the address:xiyou
will you go on?(1/0):1
please input a name:
```

第10章 用户界面设计



10.1 本章要求

- 了解用户界面。
- 掌握 tkinter。
- 掌握 wxPython。

10.2 本章知识重点

10.2.1 界面设计原则

界面设计一般应遵循如下原则:

(1) 界面具有一致性。一致性原则在界面设计中最容易被忽视,同时也最容易修改。例如,在菜单和联机帮助中必须使用相同的术语,对话框必须具有相同的风格,等等。

(2) 常用操作设置快捷键。常用操作的使用频度大,应该减少操作序列的长度。例如,设置常用操作(如打开、存盘、另存等)的快捷键。

(3) 提供简单错误处理。系统对于错误能进行检测,并提供快速简单的处理办法。

(4) 提供信息反馈。对常用操作和简单操作的反馈可以不做要求,但是对不常用操作和至关重要的操作应提供信息的反馈。

(5) 操作可逆。可逆的动作可以是单个的操作,也可以是相对独立的操作序列。

(6) 联机帮助。对于不熟练的用户来说,良好的联机帮助具有非常重要的作用。

10.2.2 wxPython 开发流程

在网址 http://wxpython.org/download.php 下载 wxPython3.0-win32-py27 文件,双击安装,默认安装路径为 C:\Python27\Lib\site-packages。

wxPython 开发流程包括如下步骤:

(1) 导入必须的 wxPython 包。

(2) 子类化 wxPython 应用程序类。

（3）定义一个应用程序的初始化方法。

（4）创建一个应用程序类的实例。

（5）进入这个应用程序的主事件循环。

【例 10-1】　最小的空的 wxPython 程序。

```
# coding=utf-8
import wx                          # 导入必须的 Python 包
class App(wx.App):                 # 子类化 wxPython 应用程序类
    def OnInit(self):              # 定义一个应用程序的初始化方法
        frame=wx.Frame(parent=None, title='')
        frame.Show()
        return True
app=App()                          # 创建一个应用程序类的实例
app.MainLoop()                     # 进入这个应用程序的主事件循环
```

程序运行结果如图 10.1 所示。



图 10.1　例 10-1 程序运行结果

10.2.3　事件处理

事件处理是 wxPython 程序工作的基本机制，首先了解如下几个术语：

- 事件（event）是每一个 GUI 应用程序期间发生的事情，要求有一个响应。
- 事件对象（event object）代表具体一个事件，为 wx.Event 或其子类的实例，如 wx.CommandEvent、wx.MouseEvent。
- 事件类型（event type）是一个已产生的独特的事件，wxPython 分配给每个事件对象的一个整数 ID。
- 事件绑定是一个对象，结合事件处理程序的事件类型。
- 事件循环是一种编程构造和调度事件或程序中的消息等待。事件对象是与事件相关联的对象。它通常是一个窗口。

在 wxPython 中，事件绑定的语法如下：

```
Bind(event, handler, source=None, id=wx.ID_ANY, id2=wx.ID_ANY)?
```

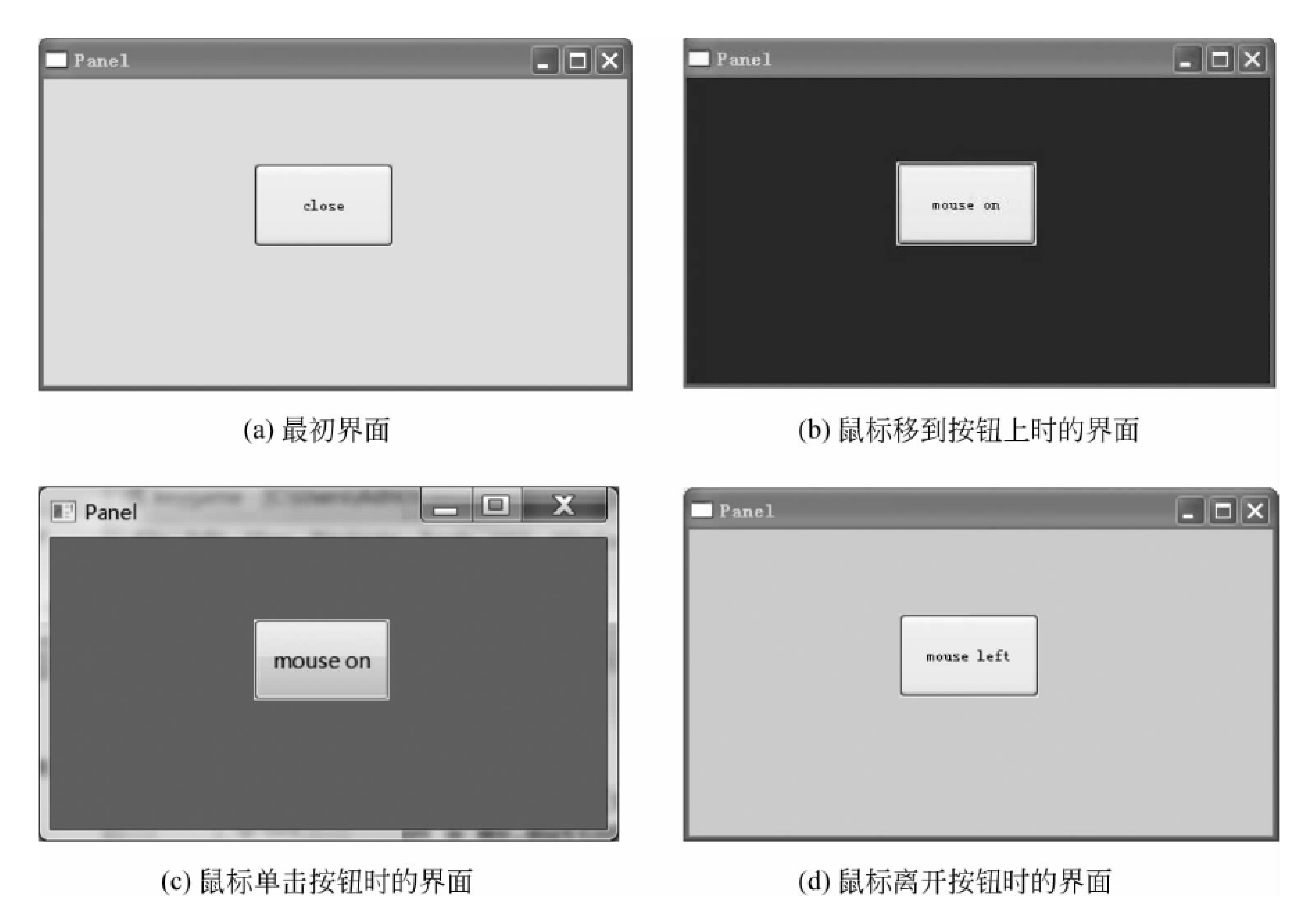
参数说明：

- event，EVT__* 对象。
- handler，方法。
- source，用于从不同的小部件中区分相同的事件类型。
- id，用于区分多个按钮、菜单项等。

【例 10-2】 实现鼠标移动功能。

```
import wx
class MyFrame(wx.Frame):
    def __init__(self, parent, id):
        wx.Frame.__init__(self, parent, id, 'Panel', size=(600, 300))
        self.panel=wx.Panel(self)
        self.button=wx.Button(self.panel, label='close', pos=(150, 60),
            size=(100, 60))
        self.Bind(wx.EVT_BUTTON, self.OnCloseMe, self.button)
        self.button.Bind(wx.EVT_ENTER_WINDOW, self.OnEnterWindows)
        self.button.Bind(wx.EVT_LEAVE_WINDOW, self.OnLeaveWindows)
    def OnCloseMe(self, event):
        self.panel.SetBackgroundColour('Red')
        self.panel.Refresh()
    def OnEnterWindows(self, event):
        self.panel.SetBackgroundColour('Blue')
        self.panel.Refresh()
        self.button.SetLabel("mouse on")
        event.Skip()
    def OnLeaveWindows(self, event):
        self.panel.SetBackgroundColour('Green')
        self.panel.Refresh()
        self.button.SetLabel("mouse left")
        event.Skip()
if __name__=='__main__':
    app=wx.PySimpleApp()
    frame=MyFrame(parent=None, id=-1)
    frame.Show()
    app.MainLoop()
```

运行该程序，最初界面如图 10.2(a)所示；当鼠标移动到按钮上时，界面如图 10.2(b)所示；当鼠标单击按钮时，界面如图 10.2(c)所示；当鼠标从按钮上离开时，界面如图 10.2(d)所示。



(a) 最初界面　　(b) 鼠标移到按钮上时的界面

(c) 鼠标单击按钮时的界面　　(d) 鼠标离开按钮时的界面

图 10.2　例 10-2 程序运行结果

10.3　课后习题答案

1. 建立应用程序，窗体如图 10.3 所示，有 1 个简单组合框、4 个命令按钮、1 个文本框和 1 个标签。要求如下：

(1) 单击“添加”按钮可将输入的内容添加到组合框中。

(2) 单击“删除”按钮可删除组合框中选定的项目。

(3) 单击“统计人数”按钮，可将组合框中的项目总数输出到下面的“人数：”文本框。

(4) 单击“结束”按钮或按 Esc 键退出程序。

图 10.3　程序运行界面

【解答】

```
# coding= utf- 8
import wx
class MyFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None,- 1, " Example", size= (400, 400))
        panel= wx.Panel(self,- 1)
        wx.StaticText(panel,- 1, "input name", (100, 50))
        sampleList= ['three', 'four', 'five', 'six', 'seven']
        wx.ComboBox(panel,- 1, "", (100, 100), wx.DefaultSize, sampleList, wx.CB_SIMPLE)
        wx.StaticText(panel,- 1, "count of people", (100, 280))
```

```
        self.posCtrl=wx.TextCtrl(panel,-1, "", pos=(100, 300)) #输入文本
        self.btnAdd=wx.Button(panel,-1,label="add",pos=(250,100),size=(100,30))
        self.btnDel=wx.Button(panel,-1,label="del",pos=(250,150),size=(100,30))
        self.btnCount=wx.Button(panel,-1,label="Count",pos=(250,200),size=(100,30))
        self.btnEnd=wx.Button(panel,-1,label="end",pos=(250,250),size=(100,30))
        #单击 btnEnd按钮会调用 OnEndMe函数
        self.Bind(wx.EVT_BUTTON,self.OnEndMe,self.btnEnd)
        #将 btnAdd的单击事件与 Add函数绑定
        #self.Bind(wx.EVT_BUTTON,self.Add,self.btnAdd)
    #def Add(self,event):
        #self.posCtrl.SetValue("%s" %wx.ComboBox.GetItems)
        #将 btnCount 的单击事件与 count 函数绑定
        self.Bind(wx.EVT_BUTTON,self.count,self.btnCount)
    def count(self,event):
        self.posCtrl.SetValue("%s" %wx.ComboBox.GetCount)
    def OnEndMe(self,event):
        self.Destroy()
if __name__=='__main__':
    app=wx.App()
    frame=MyFrame()
    frame.Show()
    app.MainLoop()
```

10.4 习题与解答

10.4.1 习题

1. 计算 1+2+3+…+n,如图 10.4 所示,在小窗口中输入 n 的值,单击 compute 按钮,输出结果。

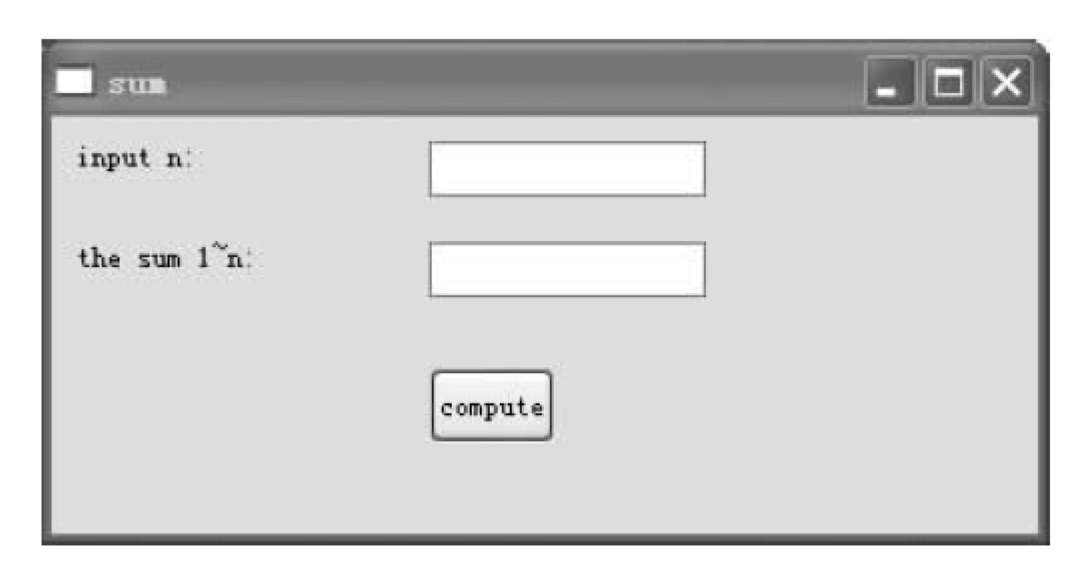


图 10.4 题 1 程序运行界面

2. 实现简单绘图功能,如图 10.5 所示,用鼠标作为笔,在窗体上进行书写。

3. 关闭窗口时,显示"Are you Sure to Quit?"提示信息,由用户确认是否真的退出程序,如图 10.6 所示。

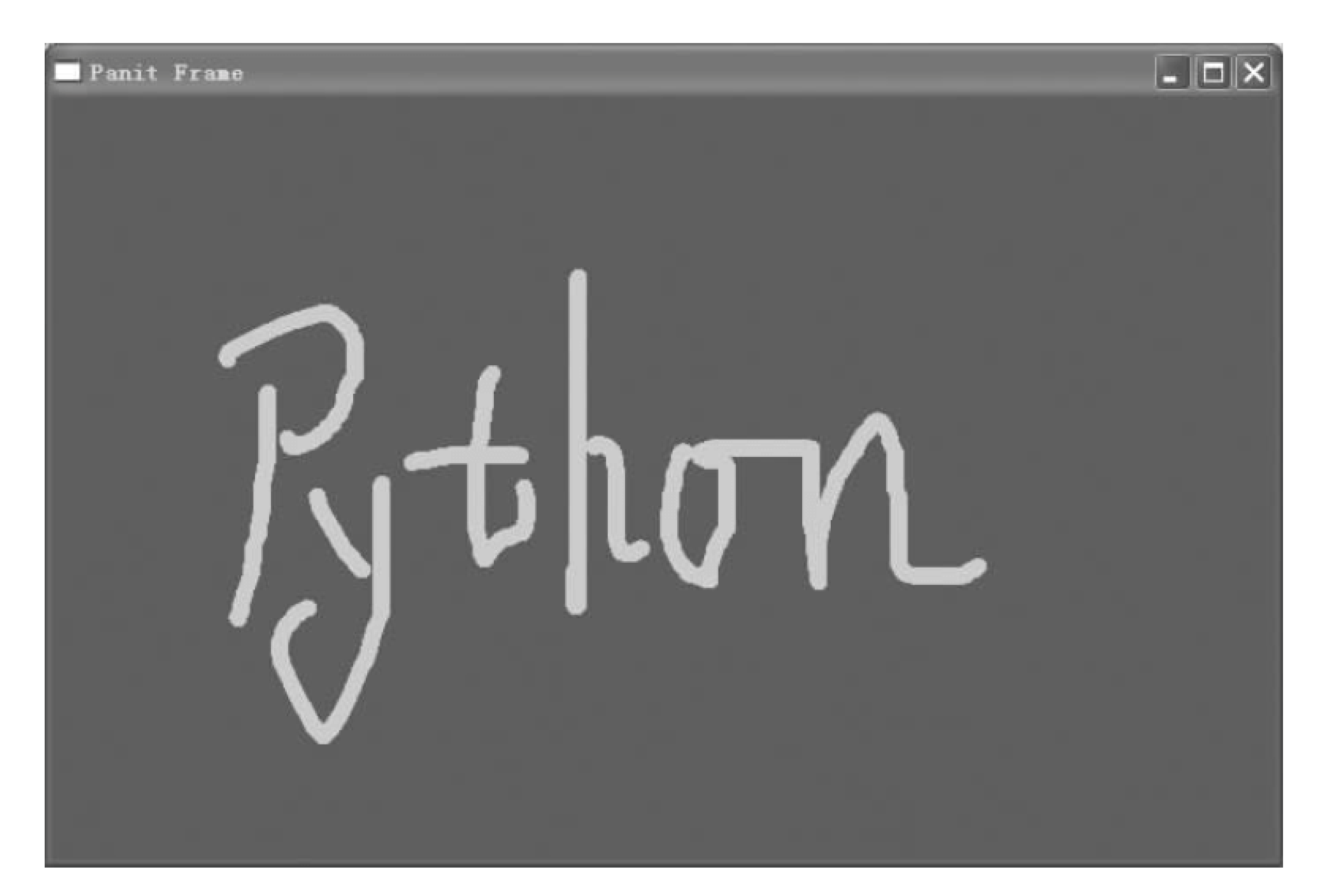


图 10.5　题 3 程序运行界面

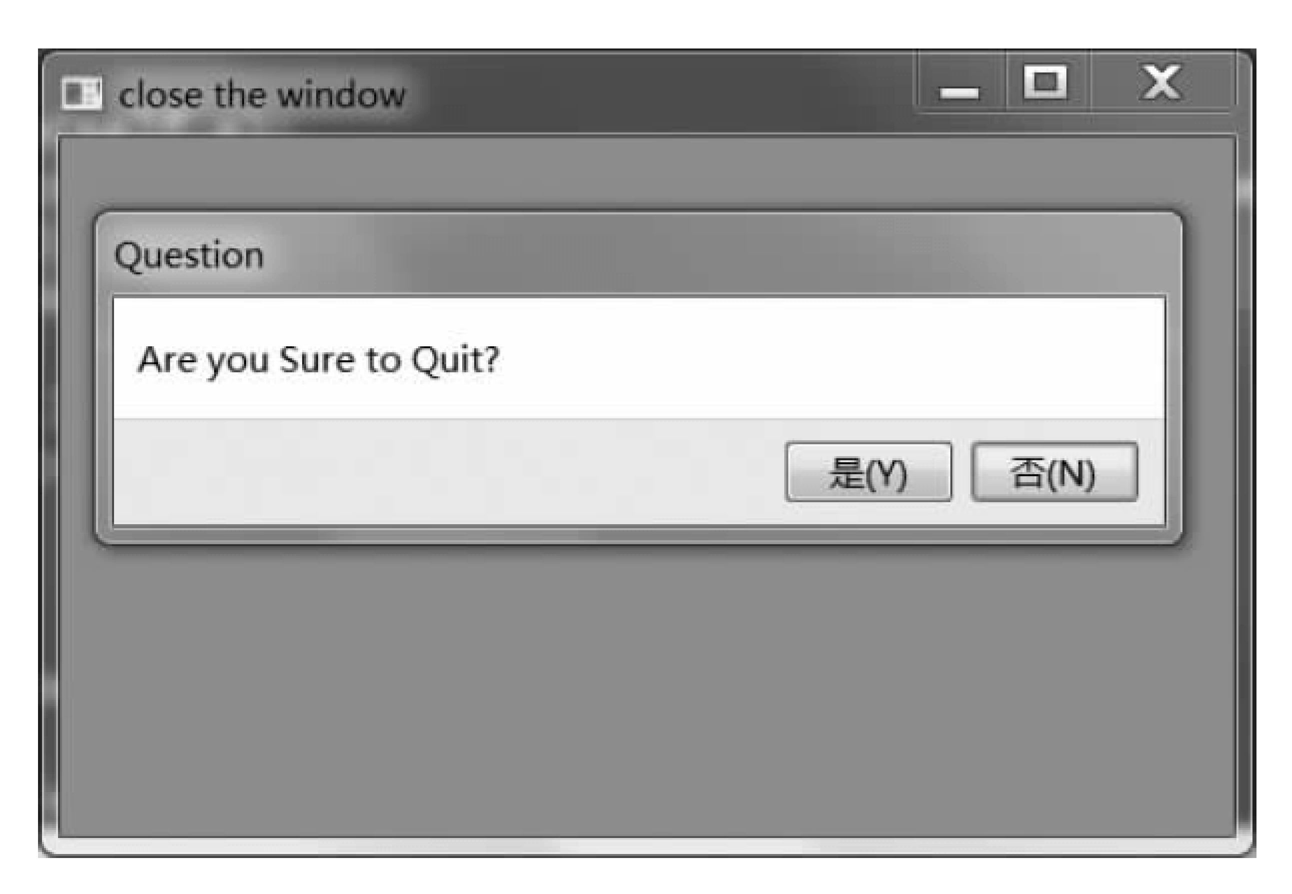


图 10.6　界面

10.4.2　习题参考答案

1. 计算 1+2+3+…+n，在小窗口中输入 n 的值，单击 compute 按钮，输出结果。

【解答】

```
import wx
class Frame(wx.Frame):
    def __init__(self,superion):
        wx.Frame.__init__(self,parent=superion,title='sum',size=(400,200))
        panel=wx.Panel(self)
        wx.StaticText(panel,label='input n:',pos=(10,10))
```

```
        self.inputN=wx.TextCtrl(panel,pos=(150,10))
        wx.StaticText(panel,label='the sum 1~ n:',pos=(10,50))
        self.outsum=wx.TextCtrl(panel,pos=(150,50))
        self.btnsum=wx.Button(panel,label='compute',pos=(150,100),size=(50,30))
        self.Bind(wx.EVT_BUTTON,self.f,self.btnsum)
    def f(self,event):
        n=self.inputN.GetValue()
        n=int(n)
        i=1
        s=0
        for i in range(1,n+1):
            s=s+i
            self.outsum.SetValue(str(s))
if __name__=='__main__':
    app=wx.App()
    frame=Frame(None)
    frame.Show()
    app.MainLoop()
```

运行程序,输入100,可得1～100的累加和,如图10.7所示。

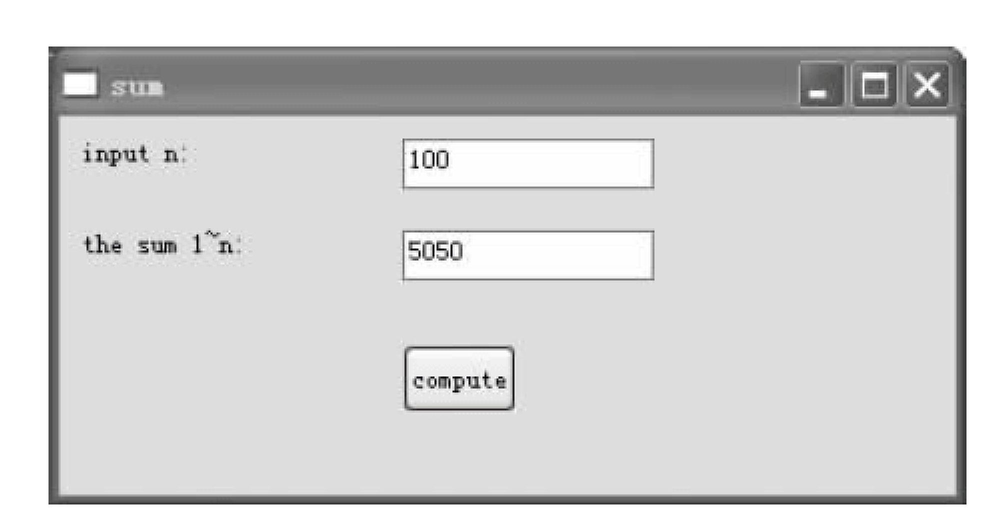


图10.7 程序运行结果

2. 实现简单绘图功能,用鼠标作为笔,在窗体上进行书写。

【解答】

```
import wx
class PaintWindow(wx.Window):
    def __init__(self, parent, id):
        wx.Window.__init__(self, parent, id)
        self.SetBackgroundColour("Red")
        self.color="Green"
        self.thickness=10
        #创建一个画笔
        self.pen=wx.Pen(self.color, self.thickness, wx.SOLID)
        self.lines=[]
        self.curLine=[]
```

```
        self.pos= (0, 0)
        self.InitBuffer()
        #连接事件
        self.Bind(wx.EVT_LEFT_DOWN, self.OnLeftDown)
        self.Bind(wx.EVT_LEFT_UP, self.OnLeftUp)
        self.Bind(wx.EVT_MOTION, self.OnMotion)
        self.Bind(wx.EVT_SIZE, self.OnSize)
        self.Bind(wx.EVT_IDLE, self.OnIdle)
        self.Bind(wx.EVT_PAINT, self.OnPaint)
    def InitBuffer(self):
        size= self.GetClientSize()
        #创建缓存的设备上下文
        self.buffer= wx.EmptyBitmap(size.width, size.height)
        dc= wx.BufferedDC(None, self.buffer)
        #使用设备上下文
        dc.SetBackground(wx.Brush(self.GetBackgroundColour()))
        dc.Clear()
        self.DrawLines(dc)
        self.reInitBuffer= False
    def GetLinesData(self):
        return self.lines[:]
    def SetLinesData(self, lines):
        self.lines= lines[:]
        self.InitBuffer()
        self.Refresh()
    def OnLeftDown(self, event):
        self.curLine= []
        #获取鼠标位置
        self.pos= event.GetPositionTuple()
        self.CaptureMouse()
    def OnLeftUp(self, event):
        if self.HasCapture():
            self.lines.append((self.color, self.thickness, self.curLine))
            self.curLine= []
            self.ReleaseMouse()
    def OnMotion(self, event):
        if event.Dragging() and event.LeftIsDown():
            dc= wx.BufferedDC(wx.ClientDC(self), self.buffer)
            self.drawMotion(dc, event)
        event.Skip()
    def drawMotion(self, dc, event):
        dc.SetPen(self.pen)
        newPos= event.GetPositionTuple()
        coords= self.pos + newPos
```

```
            self.curLine.append(coords)
            dc.DrawLine(* coords)
            self.pos= newPos
        def OnSize(self, event):
            self.reInitBuffer= True
        def OnIdle(self, event):
            if self.reInitBuffer:
                self.InitBuffer()
                self.Refresh(False)
        def OnPaint(self, event):
            dc= wx.BufferedPaintDC(self, self.buffer)
        def DrawLines(self, dc):
            for colour, thickness, line in self.lines:
                pen= wx.Pen(colour, thickness, wx.SOLID)
                dc.SetPen(pen)
                for coords in line:
                    dc.DrawLine(* coords)
        def SetColor(self, color):
            self.color= color
            self.pen= wx.Pen(self.color, self.thickness, wx.SOLID)
        def SetThickness(self, num):
            self.thickness= num
            self.pen= wx.Pen(self.color, self.thickness, wx.SOLID)
    class PaintFrame(wx.Frame):
        def __init__(self, parent):
            wx.Frame.__init__(self, parent,-1, "Panit Frame", size= (800, 600))
            self.paint= PaintWindow(self,-1)
    if __name__=='__main__':
        app= wx.PySimpleApp()
        frame= PaintFrame(None)
        frame.Show(True)
        app.MainLoop()
```

3. 关闭窗口时，显示“Are you Sure to Quit?”提示信息，由用户确认是否真的退出程序。

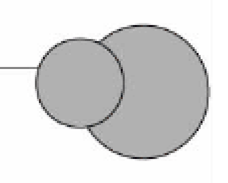
【解答】

```
import wx
class Example(wx.Frame):
    def __init__(self,parent,title):
        super(Example,self).__init__(parent,title= title,size= (400,300))
        self.InitUI()
        self.Centre()
        self.Show()
    def InitUI(self):
```

```
        self.Bind(wx.EVT_CLOSE, self.OnCloseWindow)
    def OnCloseWindow(self,e):
        dial=wx.MessageDialog(None,"Are you Sure to Quit?","Question",
            wx.YES_NO|wx.NO_DEFAULT|wx.ICON_QUESTION)
        ret=dial.ShowModal()
        if ret==wx.ID_YES:
            self.Destroy()
        else:
            e.Veto() #Veto()方法可以返回正在处理的事件
if __name__=='__main__':
    app=wx.App()
    Example(None,title="close the window")
    app.MainLoop()
```

第 11 章

绘图与科学计算



11.1　本章要求

- 了解绘图功能。
- 掌握 turtle 绘图。
- 掌握 Canvas 绘图。
- 掌握 Matplotlib。
- 掌握 NumPy。
- 掌握 scipy。
- 掌握 pandas。

11.2　本章知识重点

11.2.1　NumPy

NumPy(Numeric Python)是 Python 的开源数字扩展,用来存储和处理大型矩阵,比 Python 自身的嵌套列表结构高效。NumPy 提供了许多高级的数值编程工具,如矩阵数据类型、矢量处理等。NumPy 下载网址为 http://sourceforge.net/projects/numpy/files。

NumPy 的数组类作为实现矩阵类的基础,其存取的速度比存取 Python 列表速度快许多。另外,NumPy 是数组语言,不需要大多数循环。

【例 11-1】 NumPy 与普通 Python 的对比。

首先,用 Python 实验数值:

```
a= range(10000000)
b= range(10000000)
c= []
for i in range(len(a)):
c.append(a[i]+b[i])
```

这个循环将在 GHz 级的处理器上耗时 5～10s。

用 NumPy 实现上面的功能,代码如下:

```
import numpy as np
a=np.arange(10000000)
b=np.arange(10000000)
c=a+b
```

由于 Python 是动态类型的解释语言，意味着在每次循环迭代时都必须检查运算对象 a 和 b 的类型来选择“+”的正确意义。而当“+”的操作对象之一是 NumPy 数组时，NumPy 的 add 函数将被 Python 自动选择，仅仅检测一次类型。所以，NumPy 的导入都比普通 Python 中的循环更快。

【例 11-2】 NumPy 举例。

```
>>>import numpy as np
>>>print np.zeros((3,4))
[[ 0. 0. 0. 0.]
 [ 0. 0. 0. 0.]
 [ 0. 0. 0. 0.]]
>>>print np.ones((3,4))
[[ 1. 1. 1. 1.]
 [ 1. 1. 1. 1.]
 [ 1. 1. 1. 1.]]
>>>print np.eye(3)
[[ 1. 0. 0.]
 [ 0. 1. 0.]
 [ 0. 0. 1.]]
```

11.2.2　Matplotlib

Matplotlib 是 Python 二维绘图领域使用最广泛的图形框架，可以绘制多种形式的图形，包括线图、直方图、饼图、散点图以及误差线图等，较好地支持 TeX 排版命令。Matplotlib 类似于 MATLAB 和 R 语言，大部分函数与 MATLAB 函数同名且使用方法一致。

Matplotlib 使用 NumPy 模块提供的矩阵运算和各种数学函数，将 NumPy 统计计算结果可视化。Matplotlib 下载网址为 http://matplotlib.org/。

11.2.3　scipy

scipy 是一款方便、易于使用、专为科学和工程设计的 Python 工具包，包括统计、优化、整合、线性代数模块、傅里叶变换、信号和图像处理、常微分方程求解器等功能。在网址 http://scipy.org 下载 scipy-0.11.0rc2-win32-superpack-python2.7.exe 文件，双击安装，界面如图 11.1 所示。

scipy 库建立在 NumPy 库之上，提供了大量科学算法，主要包括如下内容：

- 特殊函数(scipy.special)。
- 积分(scipy.integrate)。

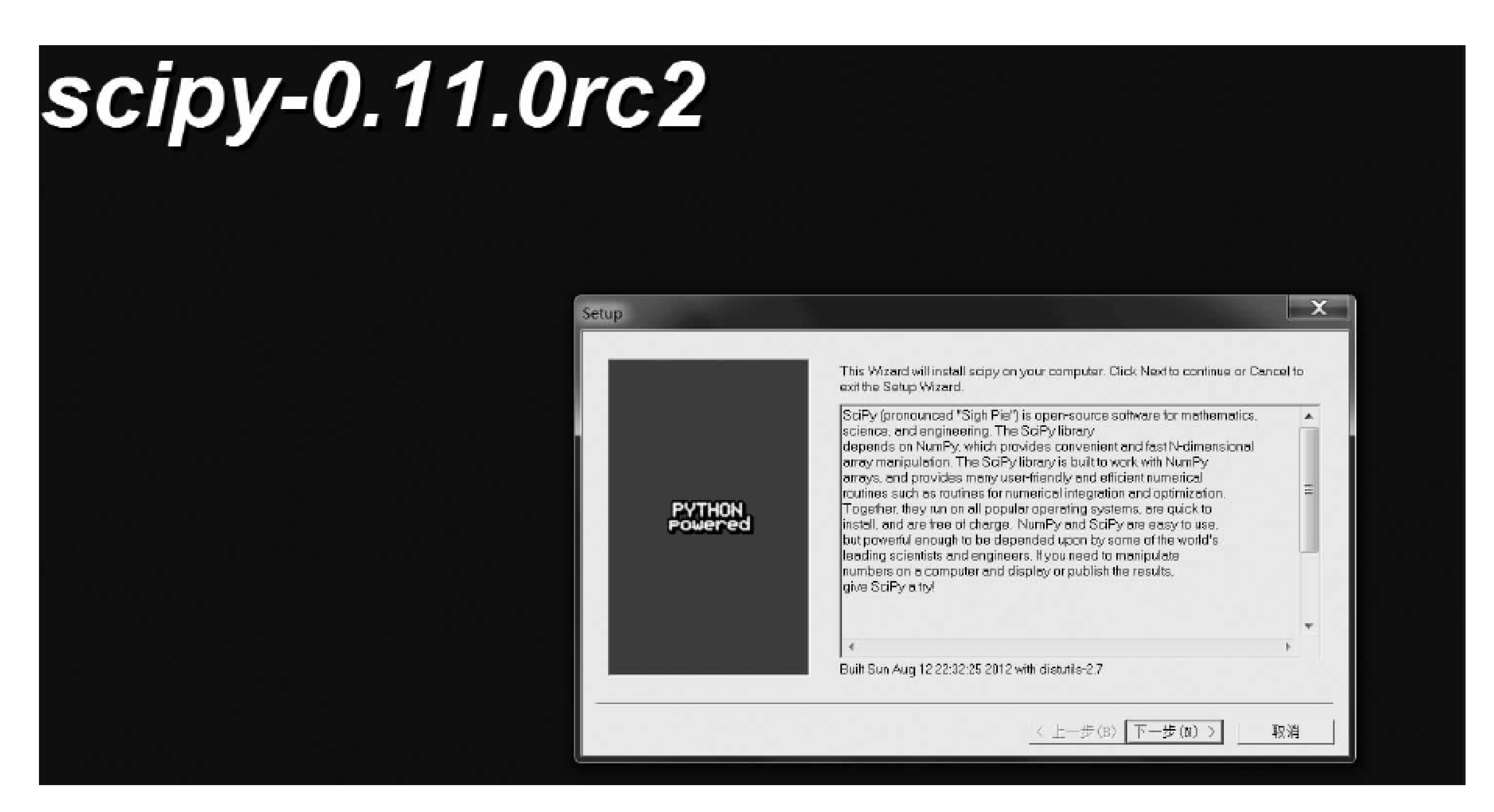


图 11.1　NumPy 下载安装

- 最优化(scipy.optimize)。
- 插值(scipy.interpolate)。
- 傅里叶变换(scipy.fftpack)。
- 信号处理(scipy.signal)。
- 线性代数(scipy.linalg)。
- 稀疏特征值(scipy.sparse)。
- 统计(scipy.stats)。
- 多维图像处理(scipy.ndimage)。
- 文件 I/O(scipy.io)。

【例 11-3】　scipy 举例。

在 C:\Python27 下创建 test.txt 文件，内容如图 11.2 所示。

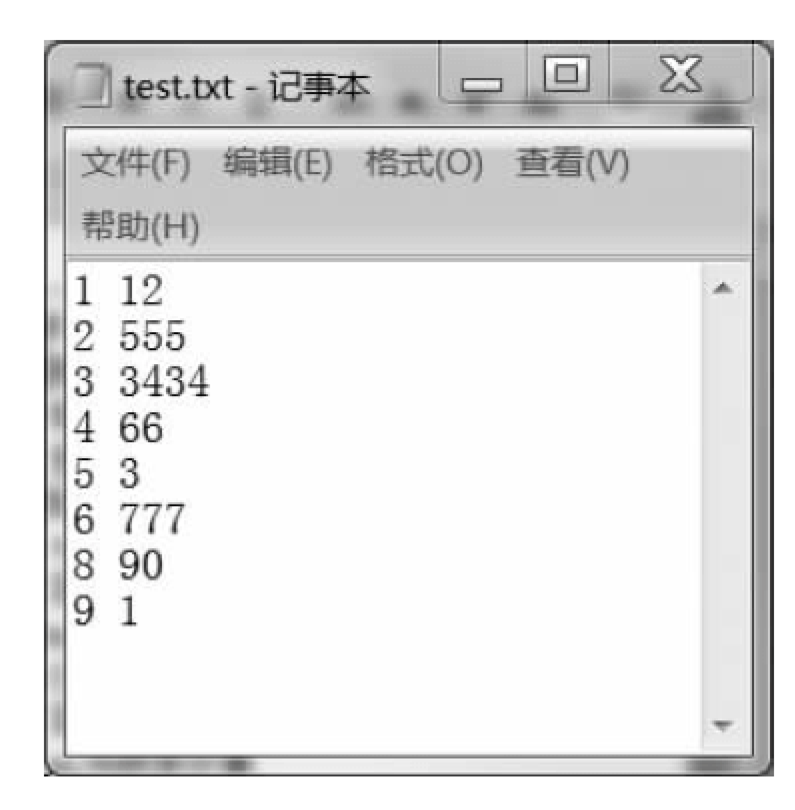


图 11.2　test.txt 文件

```
>>> import scipy as sp
>>> data= sp.genfromtxt("test.txt")
>>> print(data[:10])
[[ 1.00000000e+ 00   1.20000000e+ 01]
 [ 2.00000000e+ 00   5.55000000e+ 02]
 [ 3.00000000e+ 00   3.43400000e+ 03]
 [ 4.00000000e+ 00   6.60000000e+ 01]
 [ 5.00000000e+ 00   3.00000000e+ 00]
 [ 6.00000000e+ 00   7.77000000e+ 02]
 [ 8.00000000e+ 00   9.00000000e+ 01]
 [ 9.00000000e+ 00   1.00000000e+ 00]]
>>> print(data.shape)
(8, 2)                  #表示一共有 8行数据,每行数据有两个属性
```

```
>>>x=data[:,0]
>>>y=data[:,1]            #数据分成两个向量 x 和 y
>>>sp.sum(y)
4938.0
```

11.2.4　pandas

pandas 是基于 NumPy 的数据分析工具，具有如下两个核心数据结构：

- Series。一维的类似的数组对象，包含一个数组的数据和一个与数组关联的数据标签(索引)，与 NumPy 中的一维 Array 类似。二者与 Python 基本的数据结构 List 也很相近，其区别是：List 中的元素可以是不同的数据类型，而 Array 和 Series 中则只允许存储相同的数据类型，从而运算效率较高。
- DataFrame。类似电子表格的数据结构，包含一个经过排序的列表集。DataFrame 有行和列的索引，可以被看作一个 Series 的字典(各个 Series 共享一个索引)。在 DataFrame 中的面向行和面向列的操作大致是对称的。

下面介绍 pandas 的两种安装方法。

方法一：下载 pandas-0.11.0.win32-py2.7.exe 文件，双击安装，界面如图 11.3 所示。

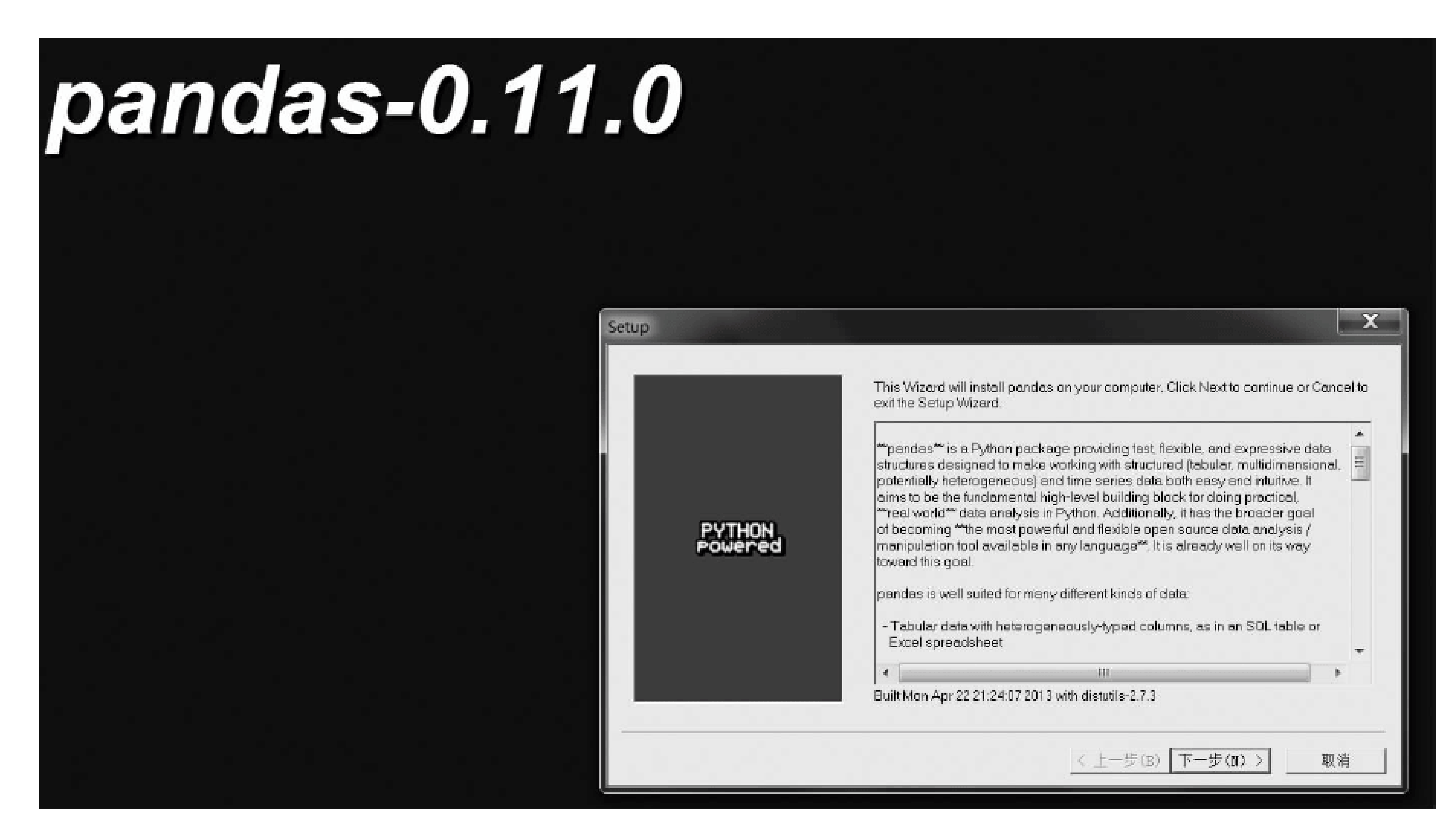


图 11.3　pandas 安装界面

配置系统环境变量。pandas 被安装到 C:\Python27\Lib\site－packages 目录下，配置环境变量如图 11.4 所示。

方法二：采用 easy_install 安装 Pandas，如图 11.5 所示。easy_install 是一个 Python 的扩展包，主要用来简化 Python 安装第三方安装包。在安装了 easy_install 之后，安装 Python 的第三方安装包就只需要在命令行中输入 easy_install packagename，然后程序会自动搜索相应版本的安装包并配置各种文件，免去了手工下载安装的复杂工作。



图 11.4　配置系统环境变量



图 11.5　采用 easy_install 安装 pandas

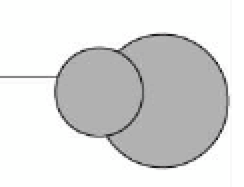
【例 11-3】 Series 举例。

```
>>> from pandas import Series,DataFrame
>>> obj= Series([4,6,-7,2])
>>> obj
0   4
1   6
2 -7
3   2
dtype: int64
>>> obj.values
array([ 4, 6,-7, 2], dtype= int64)
>>> obj.index
Int64Index([0, 1, 2, 3], dtype= int64)
```

【例 11-4】 DataFrame 举例。

```
>>> data= {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
           'year': [2000, 2001, 2002, 2001, 2002],
           'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
>>> frame= DataFrame(data)
>>> frame
   pop   state   year
0  1.5    Ohio   2000
1  1.7    Ohio   2001
2  3.6    Ohio   2002
3  2.4  Nevada   2001
4  2.9  Nevada   2002
>>> DataFrame(data, columns= ['year', 'state', 'pop'])
   year   state   pop
0  2000    Ohio   1.5
1  2001    Ohio   1.7
2  2002    Ohio   3.6
3  2001  Nevada   2.4
4  2002  Nevada   2.9
```

第 12 章



数据库应用

12.1　本章要求

- 了解关系型数据库。
- 了解 Python 数据库访问模块。
- 掌握 SQlite 访问数据库。

12.2　本章知识重点

12.2.1　关系型数据库

关系型数据库通常由一个或多个表组成。数据库中的每一个表都具有自己唯一的表名称，表由行和列组成，其中每一列包括了该列名称、数据类型以及列的其他属性等信息，行包含着某一列的记录或数据。例如，学校的学生信息就是一个二元关系，如图 12.1 所示。

name	sex	qq	tele	school
周黎明	男	9828322	88765238	西安交通大学
何明明	女	8876542	99887645	山西师范大学

图 12.1　关系型数据库的表结构

作为一个关系的二维表，必须满足以下条件：

(1) 表中每一列必须是基本数据项(即不可再分解)。

(2) 表中每一列必须具有相同的数据类型(如字符型或数值型)。

(3) 表中每一列的名字必须是唯一的。

(4) 表中不应有内容完全相同的行。

(5) 行的顺序与列的顺序不影响表格中所表示的信息的含义。

关系数据库由实体(entity)和联系(relationship)构成。实体是相互可以区别，具有一定属性的对象。联系是指实体之间的对应关系，一般分以下 3 种类型：

(1) 一对一(1∶1)。实体集 A 中每个实体至多只与实体集 B 中一个实体相联系。反之亦然。例如，班级和班主任的关系如图 12.2 所示。

(2) 一对多(1∶n)。实体集 A 中每个实体与实体集 B 中多个实体相联系，而实体集

B 中每个实体至多只与实体集 A 中一个实体相联系。例如，学生和班级的关系如图 12.3 所示。

(3) 多对多(m∶n)。实体集 A 中每个实体与实体集 B 中多个实体相联系，反之，实体集 B 中每个实体也与实体集 A 中多个实体相联系。例如，学生和课程的关系，如图 12.4 所示。

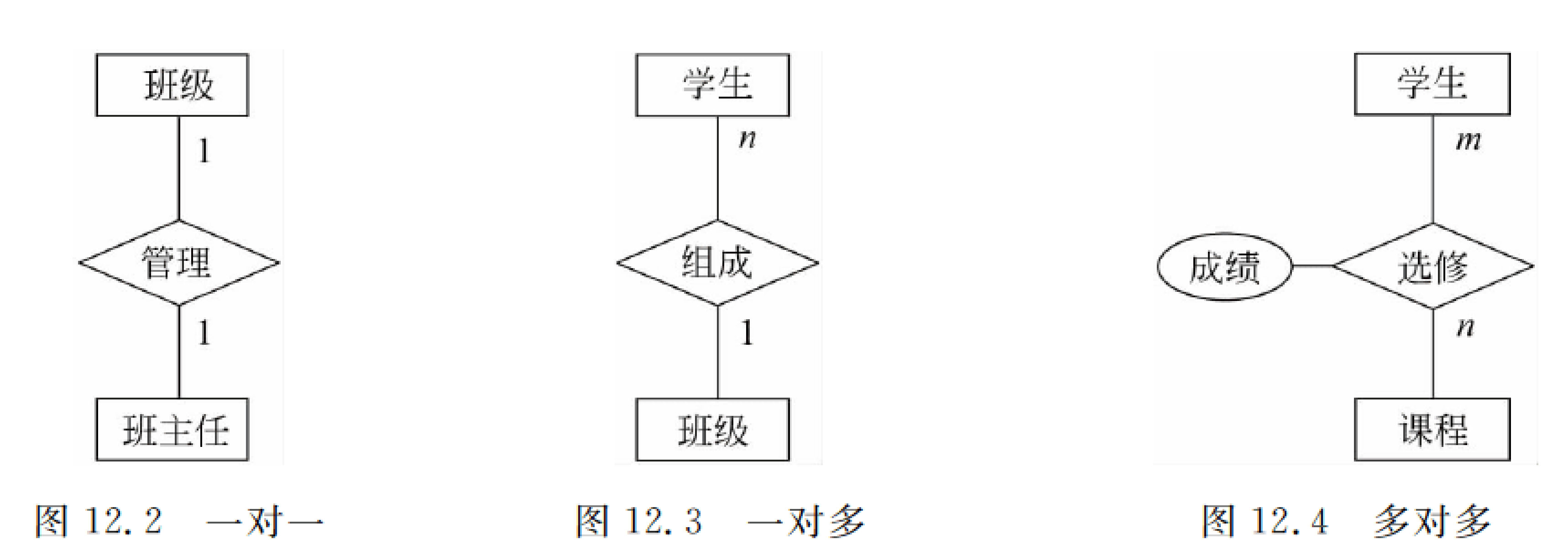


图 12.2　一对一　　图 12.3　一对多　　图 12.4　多对多

12.2.2　Python 连接数据库

下面介绍 Python 操作数据库的两个重要的概念：数据库连接对象和游标。

1. 数据库连接对象(connect)

打开数据库时返回的 conn 对象是数据库连接对象，具有以下操作：

```
conn= sqlite3.connect(host,user,passwd,db)
```

参数如下：

- host，连接的数据库服务器主机名，默认为本地主机(localhost)。
- user，连接数据库的用户名，默认为当前用户。
- passwd，连接密码，没有默认值。
- db，连接的数据库名，没有默认值。

2. 游标(cursor)

游标是数据库管理系统为用户开设的一个数据缓冲区，存放 SQL 语句的执行结果，每个游标区都有一个名字，用户可以用 SQL 语句逐一从游标中获取记录，进行操作处理。

定义一个游标的操作如下：

```
cu= conn.cursor()              # cu 为一变量
```

游标对象有以下的操作：

- execute()，执行 SQL 语句。
- executemany，执行多条 SQL 语句。
- fetchone()，从结果中取一条记录，并将游标指向下一条记录。
- fetchmany()，从结果中取多条记录。

- fetchall()，从结果中取出所有记录。
- scroll()，游标滚动。
- close()，关闭游标。

12.2.3 Python 操作数据库

Python 操作数据库的过程如下：

步骤 1：用 db.connect 创建数据库连接，返回连接对象为 conn。

步骤 2：操作数据库中的记录。

(1) 用 conn.cursor 创建游标对象 cu。

(2) 通过 cu.execute 查询数据库，用 cu.fetchall、cu.fetchone、cu.fetchmany 等方法返回查询结果。

(3) 用 conn.commit 修改数据库。

步骤 3：关闭 cu 和 conn。

12.3 课后习题答案

1. 什么是关系数据库？如何在 Access 中建立数据库，并进行增删改查操作？

【解答】 略

2. SQL 语句有哪些？如何使用？

【解答】 结构化查询语言(Structured Query Language，SQL)是操作数据库的工业标准语言，作为通用的、专门操作数据库的语言，SQL 可以确切指定想要检索的记录以及按什么顺序检索。SQL 常用命令如表 12.1 所示。

表 12.1 SQL 常用命令

命令	描述
INSERT	往数据库表中插入记录
DELETE	从数据库表中删除记录
SELECT	在数据库中查找满足特定条件的记录
UPDATE	改变特定记录和字段的值
FROM 子句	指定从中选定记录的表
WHERE 子句	指定所选记录必须满足的条件
GROUP BY 子句	把选定的记录分成特定的组
HAVING 子句	说明每个组需要满足的条件
ORDER BY 子句	按特定的次序将记录排序
AVG	获得特定字段中的值的平均数
COUNT	返回选定记录的个数
SUM	返回特定字段中所有值的总和

续表

命 令	描 述
MAX	返回指定字段中的最大值
MIN	返回指定字段中的最小值

3. Python 如何实现操作数据库？

【解答】 针对 MySQL、Oracle、DB2 等众多的数据库，Python 提供了通用数据库访问模块以及专用数据库访问模块访问各个数据库。

4. 实现例 12-5，在 Python 中操作 SQLite3。

【解答】 略

5. 下载 MySQL，实现例 12-5 的功能。

【解答】 略

6. 使用 Python 自带的轻量级 SQLite 关系型数据库，实现一个书店中对书的分类和价格信息的存储和增、删、改、查等功能。

【解答】 在 SQLite 数据库中设计 book 和 category 两个表，其中，category 表用于记录分类，book 表用于记录某个书的信息。一本书归属于某一个分类，一个分类包含多本书，两者是一对多的关系。因此 book 表有一个外键指向 catogory 表的主键 id，表的关系如图 12.5 所示。

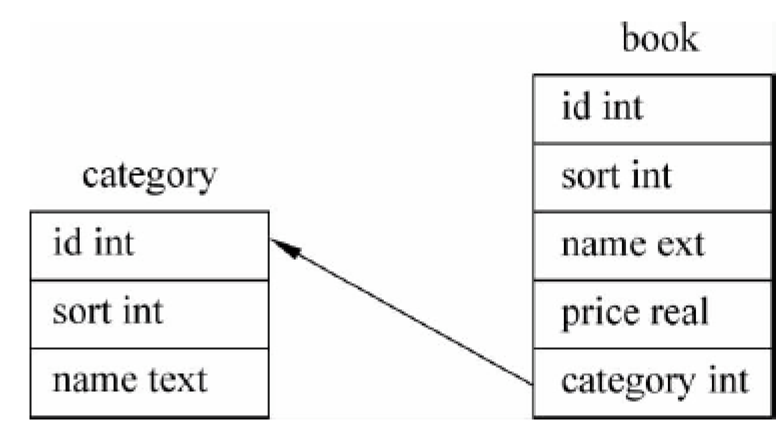


图 12.5 表的关系图

创建数据库：

```
import sqlite3
conn= sqlite3.connect("test.db")

c= conn.cursor()

# create tables
c.execute('''CREATE TABLE category(id int primary key, sort int, name text)''')
c.execute('''CREATE TABLE book
      (id int primary key,
       sort int,
       name text,
       price real,
       category int,
       FOREIGN KEY (category) REFERENCES category(id))''')

# save the changes
conn.commit()

# close the connection with the database
conn.close()
```

插入数据：

```
import sqlite3
conn= sqlite3.connect("test.db")
c= conn.cursor()

books= [(1, 1, 'Cook Recipe', 3.12, 1),
        (2, 3, 'Python Intro', 17.5, 2),
        (3, 2, 'OS Intro', 13.6, 2),
]

# execute "INSERT"
c.execute("INSERT INTO category VALUES (1, 1, 'kitchen')")
# execute multiple commands
c.executemany('INSERT INTO book VALUES (?, ?, ?, ?, ?)', books)

conn.commit()
conn.close()
```

查询数据：

```
import sqlite3
conn= sqlite3.connect('test.db')
c= conn.cursor()

# retrieve one record
c.execute('SELECT name FROM category ORDER BY sort')
print(c.fetchone())
print(c.fetchone())

# retrieve all records as a list
c.execute('SELECT * FROM book WHERE book.category= 1')
print(c.fetchall())

# iterate through the records
for row in c.execute('SELECT name, price FROM book ORDER BY sort'):
    print(row)
```

删除数据：

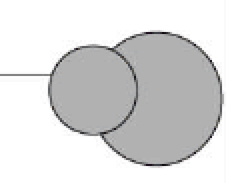
```
conn= sqlite3.connect("test.db")
c= conn.cursor()

c.execute('UPDATE book SET price= ? WHERE id= ?', (1000, 1))
c.execute('DELETE FROM book WHERE id= 2')

conn.commit()
conn.close()
```

第 13 章

网络编程



13.1 本章要求

- 了解网络编程的基本概念。
- 掌握 TCP/IP 的基本含义。
- 掌握基于 Socket 网络编程。
- 掌握基于 poplib 和 smtplib 的网络编程。

13.2 本章知识重点

13.2.1 TCP/IP 四层模型

TCP/IP 是一组用于实现网络互联的通信协议，参考模型有网络接入层、网际互联层、传输层和应用层 4 个层次。

1. 应用层

应用层对应 OSI 参考模型的应用层、表示层和会话层，为用户提供各种服务。

2. 传输层

传输层对应于 OSI 参考模型的传输层，为应用层实体提供端到端的通信功能，保证了数据包的顺序传送及数据的完整性。该层定义了 TCP 和 UDP 两个主要的协议。

1）传输控制协议（Transmission Control Protocol，TCP）

TCP 协议提供的是一种可靠的、通过“三次握手”来连接的数据传输服务，TCP 协议提供可靠通信服务，例如 HTTP、FTP、SMTP 等都使用 TCP 传输协议。

2）用户数据报协议（User Datagram Protocol，UDP）

UDP 协议提供的则是不保证可靠、无连接的数据传输服务，但其传输更简单高效，适合于实时交互性应用，如音频、视频会议等。

3. 网际互联层

网际互联层对应 OSI 参考模型的网络层，主要解决主机到主机的通信问题，包括网

际协议(IP 协议),IP 协议是网际互联层最重要的协议,提供可靠、无连接的数据报传递服务。

4. 网络接入层

网络接入层又称为网络访问层,与 OSI 参考模型中的物理层和数据链路层相对应,负责监视数据在主机和网络之间的交换。

OSI 参考模型与 TCP/IP 四层模型的对应关系如图 13.1 所示。

OSI参考模型	TCP/IP四层模型
应用层	应用层
表示层	
会话层	
传输层	传输层
网络层	网际互联层
数据链路层	网络接入层
物理层	

图 13.1 OSI 参考模型与 TCP/IP 四层模型

13.2.2 IP 地址和端口号

网络通信必须有 IP 地址和端口号两个重要的信息才能完成数据的传输。

1. IP 地址

为了区分网络中的每一台主机,Internet 采用一种全局通用的地址格式,为网络中的每一台主机分配唯一的地址(IP 地址)。IP 地址是通信主机的唯一标识地址,相当于邮件通讯中双方的邮件地址。由于 IP 地址基于数字标识,不易记忆,因此使用域名(Domain Name System,DNS)帮助人类记忆,如 www.sina.com.cn 就是新浪网的域名。

2. 端口号

一台计算机可以运行多个网络程序,如 QQ、IE 浏览器、MySQL 数据库等,每个网络程序都有唯一的端口号,用于区分不同的网络应用程序,相当于邮件通信中的收件人姓名。例如,新浪网的端口号为 80 端口。

13.2.3 Socket 编程

Socket 作为网络编程的一个抽象概念,用于描述 IP 地址和端口,表示“打开了一个网络链接”,其编程依托于客户/服务器的架构,实现客户端与服务器之间的单向“数据流通”,即客户端输入数据,发送到服务器端,服务器端生成(时间戳+数据)的封装数据回应客户端。Socket 编程有面向连接和无连接两种,分别对应 TCP 数据流和 UDP 数据报文。

Python 编写服务器端程序的步骤如下:

步骤 1. 创建 socket 对象。调用 socket 构造函数。

```
socket= socket.socket(family, type)
```

family 参数代表地址家族,可为 AF_INET 或 AF_UNIX。AF_INET 家族包括 Internet 地址,AF_UNIX 家族用于同一台机器上的进程间通信。type 参数代表套接字类型,可为 SOCK_STREAM(流套接字)和 SOCK_DGRAM(数据报套接字)。

步骤 2. 将 socket 绑定到指定地址。通过 socket 对象的 bind 方法实现：

```
socket.bind(address)
```

由 AF_INET 所创建的套接字，address 地址必须是一个双元素元组，格式是(host，port)。host 代表主机，port 代表端口号。

步骤 3. 使用 socket 套接字的 listen 方法接收连接请求：

```
socket.listen(backlog)
```

backlog 指定最多允许多少个客户连接到服务器，至少为 1。收到连接请求后，请求需要排队，如果队列满，就拒绝请求。

步骤 4. 服务器套接字通过 socket 的 accept 方法等待客户请求一个连接：

```
connection, address= socket.accept()
```

调用 accept 方法时，socket 会进入 waiting 状态。客户请求连接时，accept 方法建立连接并返回服务器。accept 方法返回一个含有两个元素的元组(connection，address)。第一个元素 connection 是新的 socket 对象，服务器必须通过它与客户通信；第二个元素 address 是客户的 Internet 地址。

步骤 5. 处理阶段，服务器和客户端通过 send 和 recv 方法传输数据。

服务器调用 send，并采用字符串形式向客户发送信息。send 方法返回已发送的字符个数。服务器使用 recv 方法从客户接收信息。调用 recv 时，服务器必须指定一个整数，它对应于可通过本次方法调用来接收的最大数据量。recv 方法在接收数据时会进入 blocked 状态，最后返回一个字符串，用它表示收到的数据。如果发送的数据量超过了 recv 所允许的大小，数据会被截短，多余的数据将缓冲于接收端。以后调用 recv 时，多余的数据会从缓冲区删除(以及自上次调用 recv 以来客户可能发送的其他任何数据)。

步骤 6. 传输结束，服务器调用 socket 的 close 方法关闭连接。

Python 编写客户端程序的步骤如下：

步骤 1. 创建一个 socket 连接服务器：

```
socket= socket.socket(family, type)
```

步骤 2. 使用 socket 的 connect 方法连接服务器。

对于 AF_INET 家族，连接格式如下：

```
socket.connect((host,port))
```

host 代表服务器主机名或 IP，port 代表服务器进程所绑定的端口号。如连接成功，客户就可通过套接字与服务器通信；如果连接失败，会引发 socket.error 异常。

步骤 3. 处理阶段，客户和服务器通过 send 方法和 recv 方法通信。

步骤 4. 传输结束，客户通过调用 socket 的 close 方法关闭连接。

【例 13-1】 基于 socket 的网络编程。

server.py 代码如下：

```
if __name__ == '__main__':
    import socket
    sock= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.bind(('localhost', 8001))
    sock.listen(5)
    while True:
        connection,address= sock.accept()
        try:
            connection.settimeout(5)
            buf= connection.recv(1024)
            if buf == '1':
                connection.send('welcome to server! ')
            else:
                connection.send('please go out! ')
        except socket.timeout:
            print 'time out'
        connection.close()
```

client. py 代码如下：

```
if __name__ == '__main__':
    import socket
    sock= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(('localhost', 8001))
    import time
    time.sleep(2)
    sock.send('1')
    print sock.recv(1024)
    sock.close()
```

在终端运行 server. py，然后运行 clien. py，会在终端打印“welcome to server!”。如果更改 client. py 的 sock. send('1')为其他值，在终端会打印“please go out!”，更改 time. sleep(2)为大于 5 的数值，服务器将会超时。

13.3　课后习题答案

1. TCP/IP 协议的四层模型是什么？

【解答】 TCP/IP 是一组用于实现网络互联的通信协议，其参考模型有网络接入层、网际互联层、传输层和应用层四个层次。

2. 如何理解 IP 地址？

【解答】 为了区分网络中的每一台主机，Internet 采用一种全局通用的地址格式，为网络中的每一台主机分配唯一的地址（IP 地址）。IP 地址是通信主机的唯一标识地址，相当于邮件通信中双方的邮件地址。

3. 端口号的作用是什么?

【解答】 同一台计算机可以运行多个网络程序,如 QQ、IE 浏览器、MySQL 数据库等,端口号的作用就是区分不同的网络应用程序,每个网络程序都有唯一的端口号,端口号相当于邮件通信中的收件人姓名。例如,新浪网的端口号为 80 端口。

4. 实现教材中基于 TCP 的 Socket 连接的举例。

【解答】 略

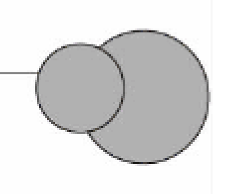
5. 实现教材中基于 UDP 的 Socket 连接的举例。

【解答】 略

6. 实现 SMTP 发送邮件和 POP3 收取邮件举例。

【解答】 略

第14章



异常处理

14.1 本章要求

- 了解错误类型。
- 掌握 Python 异常处理。
- 掌握 PyCharm 调试功能。

14.2 本章知识重点

14.2.1 错误类型

编程过程中一般会遇到语法错误、运行时错误和逻辑错误。

(1) 语法错误。语法是指语句的形式必须符合 Python 语言的要求。在编辑代码时，Python 会对输入的代码进行语法检查，不但会给出错误提示，而且标出错误位置。

(2) 运行时错误。有些代码在编写时没有错误，但在运行过程中会发生异常，这类错误称为“运行时错误”，例如执行除数为零的除法运算、打开不存在的文件、列表索引越界等。

(3) 逻辑错误。又称为语义错误，表现形式是程序运行时不报错，但结果不正确，这往往是由于程序存在逻辑上的缺陷。例如，运算符使用的不合理，语句的次序不对，循环语句的起始终值不正确等，都是逻辑错误的表现形式。对于逻辑错误，Python 解释器无能为力，只能由程序员自行发现和解决。

14.2.2 异常处理

异常(exception)是因程序的例外、违例、出错等情况而在正常控制流以外采取的行为。异常一般分为如下两个阶段：

(1) 第一个阶段是引起异常发生的错误。当一个错误发生了，异常被打印出来，称为未处理异常，这时因为没有代码来明确地关注和处理异常，异常被缺省处理，自动输出一些调试信息并终止运行。

(2) 第二个阶段是检测并处理异常。如果通过代码去明确地处理异常，则程序不会终止运行，并能增大容错性。Python 提供 try…except 语句处理异常，通过 raise 语句引发异常。

Python 的 try 语句有两种风格，一种是处理异常(try…except)，另一种是无论是否发生异常都将执行最后的代码(try…finally)。

① try…except 语句。

try…except 语句提供了异常处理机制，保护可能导致运行时错误的某些代码行。

try…except 格式如下：

```
try:
    try块                 #被监控的语句
except Exception[, reason]:
    except块              #处理异常的语句
```

try 子句中的代码块放置可能出现异常的语句，except 子句中的代码块处理异常。

② try…finally 语句。

try…finally 的用处是：无论是否发生异常，都要确保资源释放代码的执行。一般来说，如果没有发生错误，执行过 try 语句块之后执行 finally 语句块，完成整个流程。如果 try 语句块发生了异常，抛出了这个异常，会执行 except 语句块，然后运行 finally 语句块进行资源释放处理。

try…finally 格式如下：

```
try:
    try块                 #被监控的语句
except Exception[, reason]:
    except块              #处理异常的语句
finally:
    finally块
```

总之，except 语句用法如表 14.1 所示。

表 14.1 except 语句用法

分 句 形 式	说 明
except	捕获所有异常类型
except name	只捕获指定类型异常
except name,value	捕获所列异常，并获得抛出的异常对象
except (name1,name2)	捕获任何列出类型的异常
except (name1,name2),value	捕获任何列出类型的异常，并获得抛出的异常对象
else	如果没有异常发生，则运行
finally	不管有没有异常，都运行此代码块

14.2.3 PyCharm 调试功能

下面通过例子来了解 PyCharm 调试功能。

【例 14-1】 PyCharm 调试功能。

```
i=1
```

```
sum= 0
while i<=100:
    sum= sum + i
    i+=1
print "sum",sum
```

采用 PyCharm IDE 调试例 14-1，具体步骤如下。

步骤 1：设置断点。在源码中设置断点。通过单击代码左侧的空白槽来在对应位置生成断点，代码行出现粉红色，如图 14.1 所示。

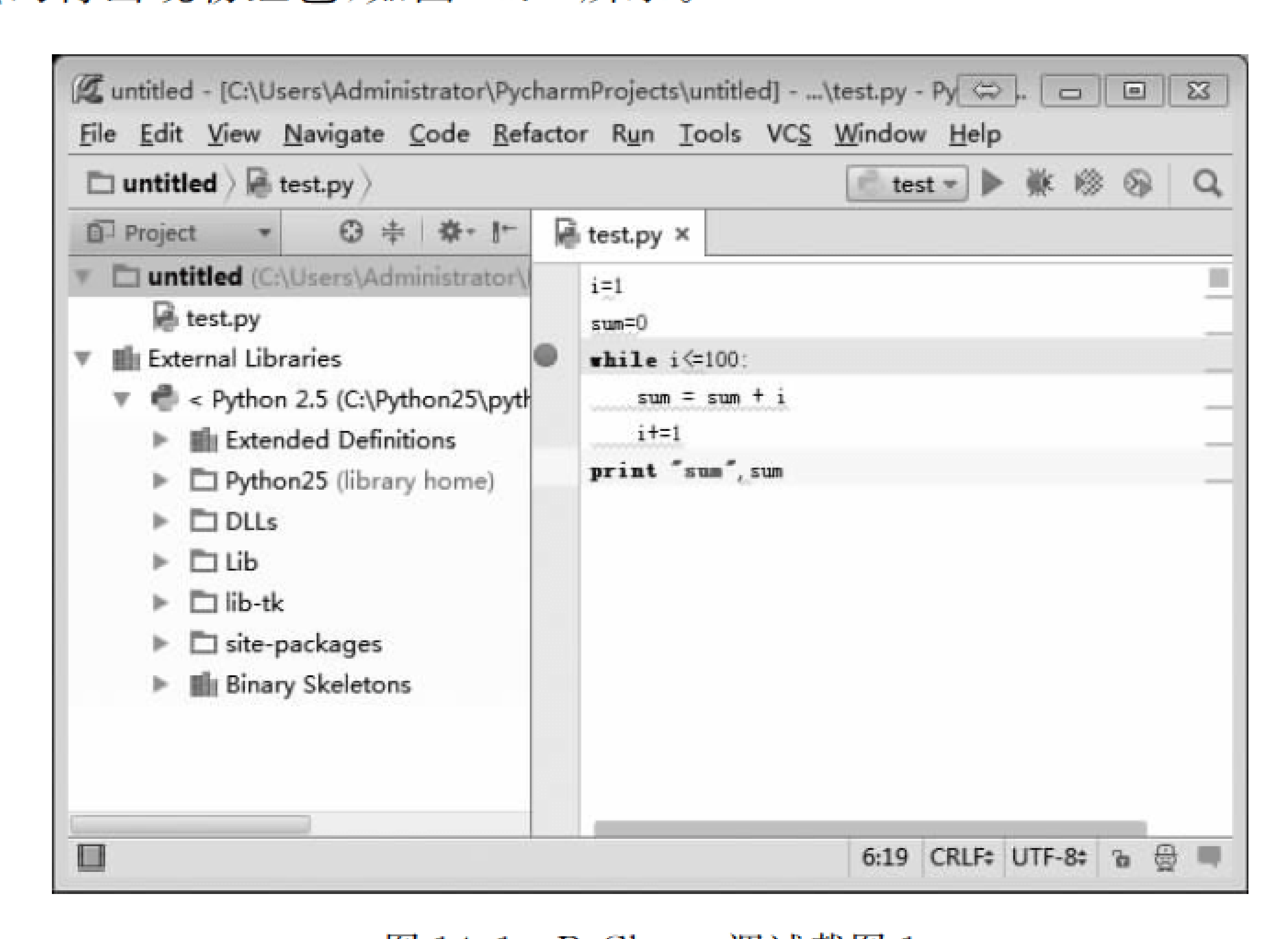


图 14.1　PyCharm 调试截图 1

步骤 2：选择 run/debug configuration ThreadSample，然后按下 Shift+F9 键（或者单击工具栏中的绿色蜘蛛形按钮），如图 14.2 所示。

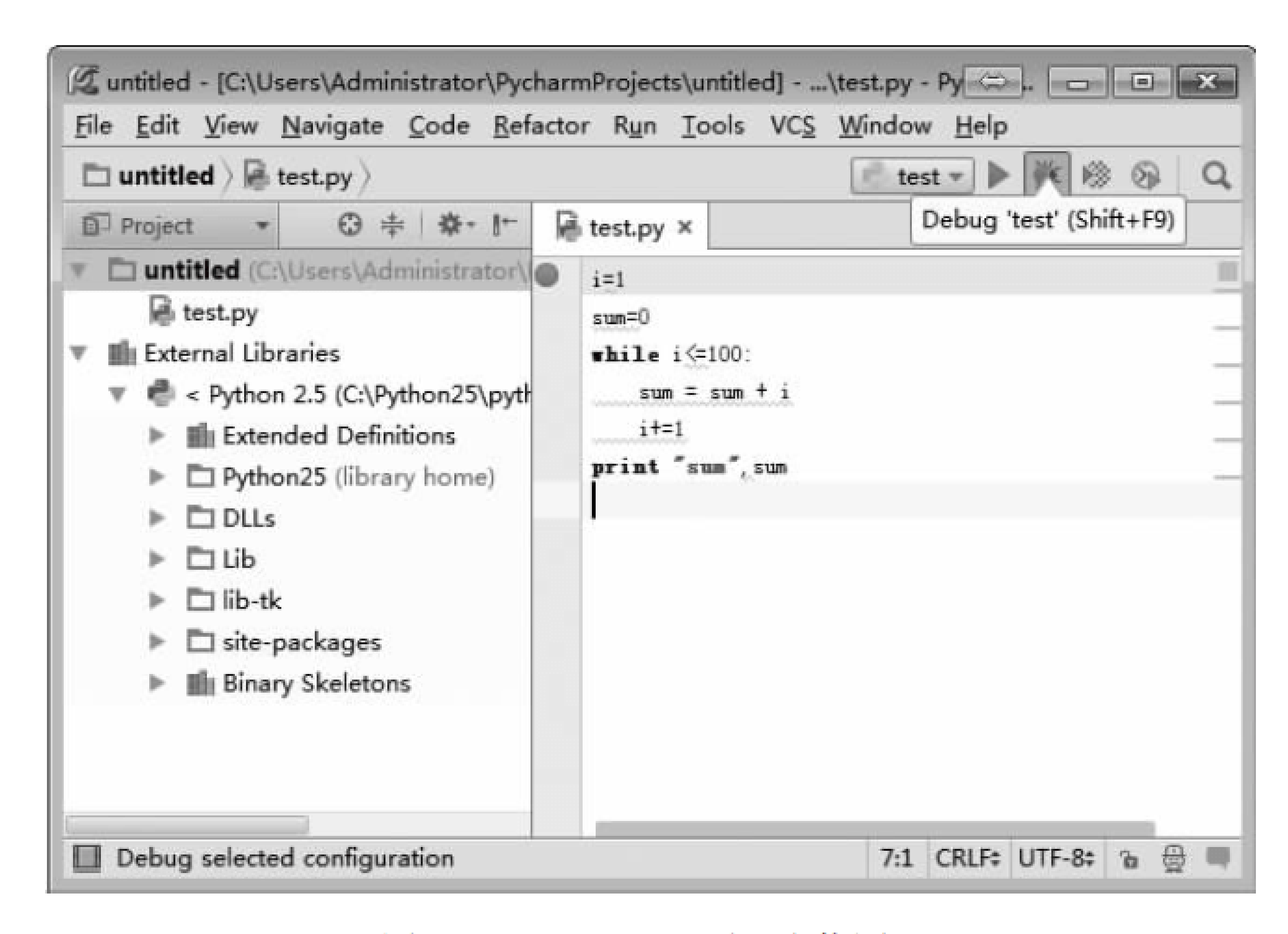


图 14.2　PyCharm 调试截图 2

调试开始，会运行到第一个断点停止，断点所在的行变为蓝色，说明 PyCharm 已经到达了这个断点，但尚未执行这行代码，如图 14.3 所示。

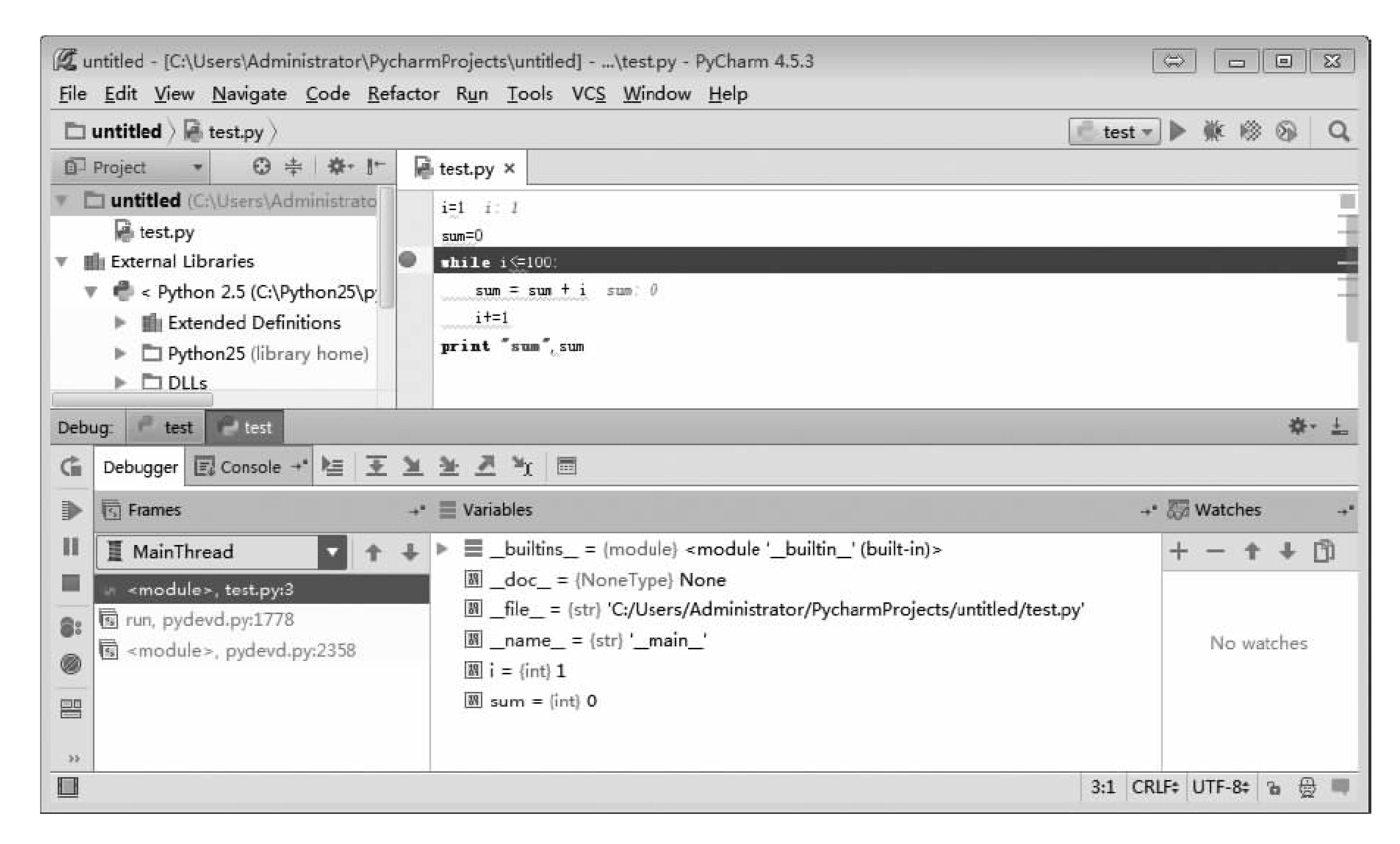


图 14.3　PyCharm 调试截图 3

步骤 3：设置一个查看器。在 Watches 窗口中，单击绿色的加号，输入期望查看的变量名称，例如输入 sum，然后回车，如图 14.4 所示。

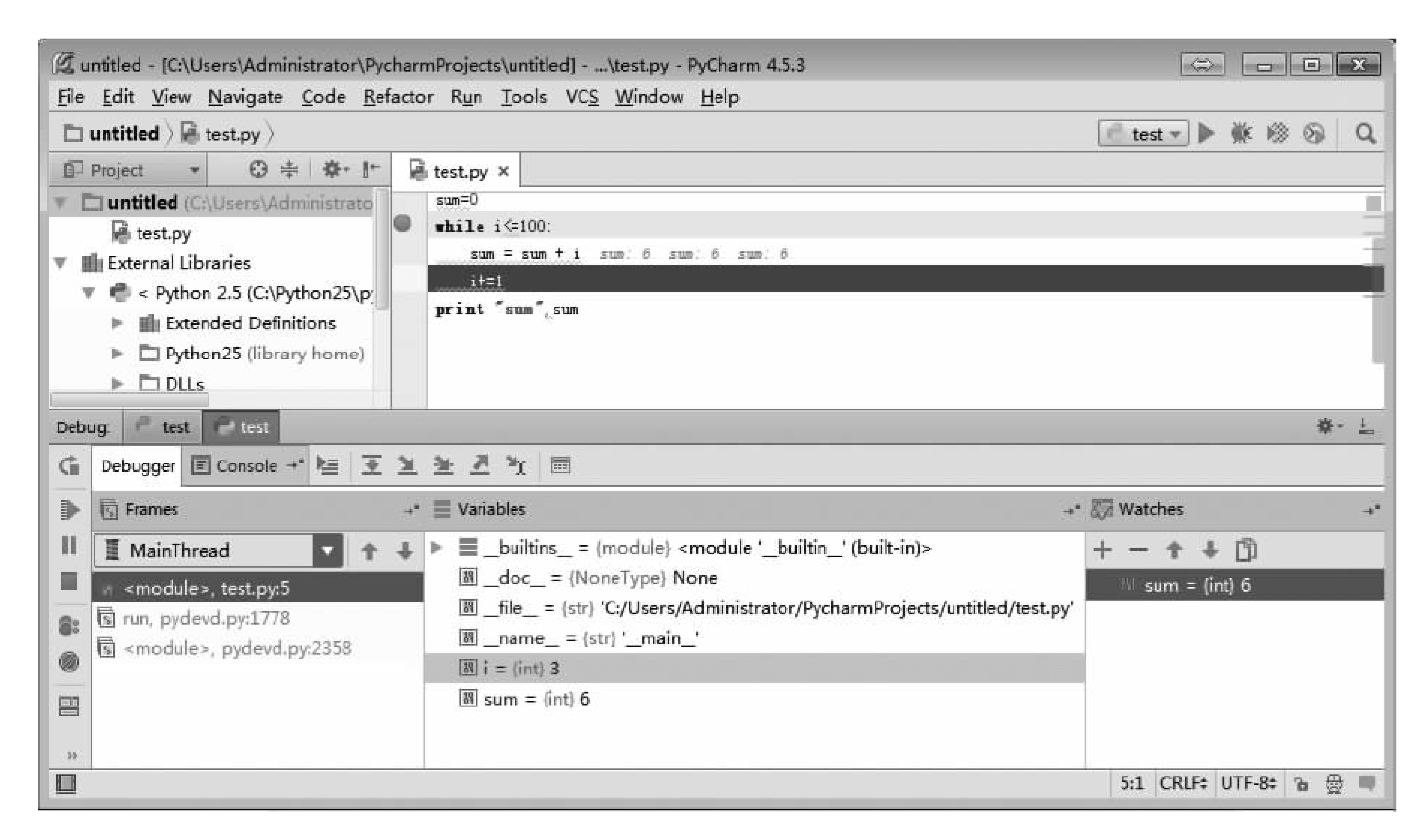


图 14.4　PyCharm 调试截图 4

步骤 4：单击 Step Over 或者按 F8 键，单步执行代码，观察变量的取值情况。

14.3 课后习题答案

1. 程序设计有几种错误？分别是什么？

【解答】 计算机编程过程中出现的错误大致分为语法错误、运行时错误和逻辑错误等。

2. 异常处理有几种？

【解答】 Python 的 try 语句有两种风格，一种是处理异常(try/except/else)，另一种是无论是否发生异常都将执行最后的代码(try/finally)。

3. 调试策略有哪些？

【解答】 调试过程的关键不是调试技术，而是用来推断错误原因的基本策略。调试的关键在于推断程序内部的错误位置及原因。可以采用以下方法：

(1) 试探法。凭经验猜测，基于错误出现的所有相关数据，假想一个错误原因，用这些数据证明或反驳它；或者一次列出所有可能的原因，通过测试一一排除。只要某次测试结果说明某种假设已初现端倪，则立即精化数据，进一步进行深入的测试。

(2) 回溯法。由错误症状最先出现的地方，沿控制流向回检查，直到找到错误根源或确定错误产生的范围。这种方法适用于小型程序。例如，程序中发现错误处是某个打印语句。通过输出值可推断程序在这一点上变量的值。再从这一点出发，回溯程序的执行过程，反复考虑："如果程序在这一点上的状态(变量的值)是这样，那么程序在上一点的状态一定是这样……"，直到找到错误的位置。

(3) 对分法。根据关键点插入的位置将程序分成两部分，分别进行调试。

(4) 归纳法。归纳法是一种从特殊推断一般的系统化思考方法。归纳法调试的基本思想是：从一些线索(错误征兆)着手，通过分析它们之间的关系来找出错误。

(5) 强行排错。这种调试方法不需要过多的思考，目前使用较多，但效率较低。

① 在程序特定部位设置打印语句，把打印语句插在出错的源程序的各个关键变量改变部位、重要分支部位、子程序调用部位，跟踪程序的执行，监视重要变量的变化。

② 自动调试工具。利用某些程序语言的调试功能或专门的交互式调试工具，分析程序的动态过程，而不必修改程序。

14.4 习题与解答

14.4.1 习题

1. 高级程序设计语言的编程 IDE 环境一般有 3 种调试工具，它们分别是什么？
2. 采用异常处理防止向文件中写数据时出错。
3. 以下是两数相加的程序：

```
x= int(input("x= "))
y= int(input("y= "))
```

```
print("x+ y= ",x+ y);
```

该程序要求接收两个整数，并输出相加结果。但如果输入的不是整数（例如输入了字母），程序就会终止执行并输出异常信息。请对程序进行修改，要求输入非整数时给出“输入内容必须为整数！”的提示，并提示用户重新输入，直至输入正确。

14.4.2 习题参考答案

1. 高级程序设计语言的 IDE 编程环境一般有 3 种调试工具，它们分别是什么？

【解答】 3 种调试工具分别是单步运行、设置断点和监视变量。将这 3 种调试工具有机地组合使用，可以帮助读者分析思考程序，找到语义错误。

(1) 单步运行。又名逐语句运行，使得程序一行一行地执行，PyCharm IDE 用粉红色光带表示程序当前的运行位置。只有单击 Step Over 或者按 F8 键，程序才能前进一行。

(2) 设置断点。程序运行到断点处就停止了，不能再往下执行了。断点是挂起程序执行的一个标记，程序执行到断点处会自动暂停，不再往下执行。可以通过单步运行和监视变量分析代码进行调试。

(3) 查看器：添加查看器用于监视变量，通过逐语句的单步执行，观察变量是如何一步一步的改变。

2. 采用异常处理防止写文件数据出错。

【解答】

```
#!/usr/bin/python
#-*-coding: UTF-8-*-
try:
    fh= open("e:\testfile", "w")
    fh.write("这是一个测试文件,用于测试异常!!")
except IOError:
    print "Error: 没有找到文件或读取文件失败"
else:
    print "内容写入文件成功"
    fh.close()
```

3. 以下是两数相加的程序：

```
x= int(input("x= "))
y= int(input("y= "))
print("x+ y= ",x+ y);
```

该程序要求接收两个整数，并输出相加结果。但如果输入的不是整数（例如输入了字母），程序就会终止执行并输出异常信息。请对程序进行修改，要求输入非整数时给出“输入内容必须为整数！”的提示，并提示用户重新输入，直至输入正确。

【解答】

```
while True:
```

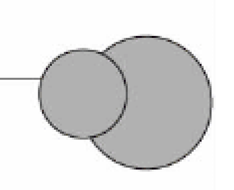
```
    try:
        x=int(input("x="))
    except ValueError:
        print("must be integer!")
    else:
        break;
while True:
    try:
        y=int(input("y="))
    except ValueError:
        print("must be integer!")
    else:
        break;
print("x+y=",x+y)
```

运行结果：

```
x="e"
must be integer!
x=1
y=4
('x+y=', 5)
```

第 15 章

网 络 爬 虫



15.1　本章要求

- 了解网络爬虫的基本概念。
- 掌握正则表达式。
- 掌握 Python re 模块。
- 掌握从网页上抓取特定信息的简单方法。

15.2　本章知识重点

15.2.1　网络爬虫简介

网络爬虫又称为网页蜘蛛或网络机器人，是一个形象的比喻：互联网是一张大网，网络爬虫就是一只蜘蛛，如果遇到资源，它就抓取下来。网络爬虫是通过一定的规则自动地抓取万维网信息的程序或者脚本。

在用户浏览网页的过程中，比如输入网址 http://www.baidu.com/，会出现百度搜索框，这是因为网址向 DNS 服务器发出一个请求，经过解析之后，向浏览器发送 HTML、JS、CSS 等文件，浏览器解析这些文件，形成百度内容。用户看到的网页实质是由爬虫抓取的由 HTML 代码构成的内容。

【例 15-1】 抓取“百度”网址。

```
import urllib2                                          #引入 urllib2 库
response=urllib2.urlopen("http://www.baidu.com")        #打开“百度”网址
print response.read()
```

【解析】 urllib2 是 Python 获取 URL(Uniform Resource Locators)的组件，通过调用 urllib2 库的 urlopen 方法传入一个 URL 地址。程序运行结果如图 15.1 所示。

15.2.2　正则表达式

正则表达式又称正规表示法、常规表示法，由普通字符和特殊字符(称为“元字符”)组成的文字模式。其中，普通字符包括没有显式指定为元字符的所有可打印和不可打印字符，包括所有大写和小写字母、所有数字、所有标点符号等。元字符则具有特殊的含义。

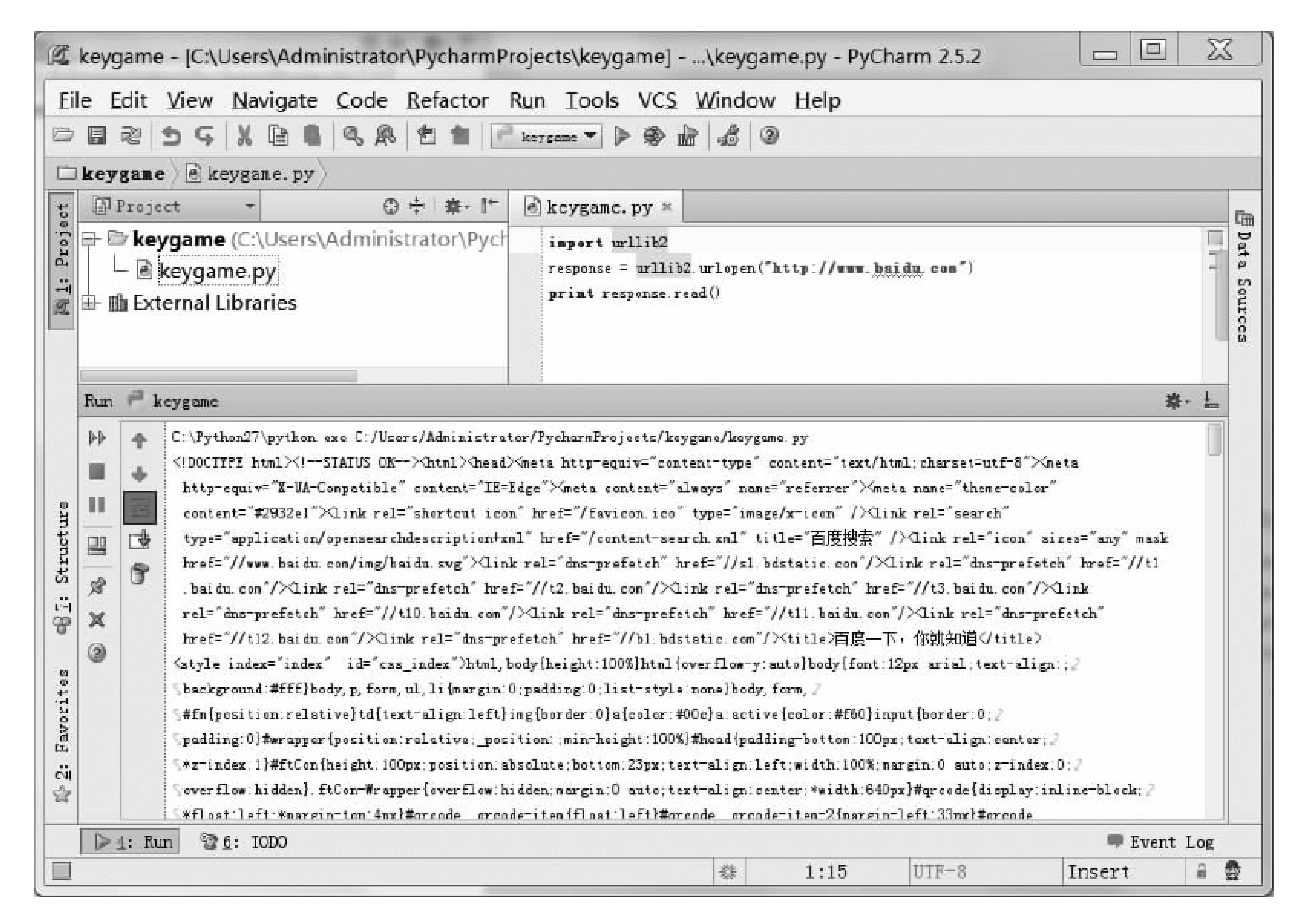


图 15.1　程序运行结果

正则表达式是对字符串操作的逻辑公式，可以实现如下功能：

- 测试字符串内的模式。例如，测试输入字符串，以查看字符串内是否出现电话号码模式或信用卡号码模式，这称为数据验证。
- 替换文本。使用正则表达式来识别文档中的特定文本，完全删除该文本或者用其他文本替换它。
- 基于模式匹配从字符串中提取子字符串，查找文档内或输入域内特定的文本。

15.2.3　Python re 模块

Python 的 re 模块提供了对正则表达式的支持。引入正则表达式 re 模块的语法如下：

```
import re
```

re 的正则表达式语法如表 15.1 所示。

表 15.1　re 的正则表达式语法

语　法	意　　义	说　　明
.	任意字符	
^	字符串开始	^hello 匹配 helloworld 而不匹配 aaaahellobbb
$	字符串结尾	hello$ 匹配 worldhello 而不匹配 aaaahellobbb
*	0 个或多个字符(贪婪匹配)	<*>匹配<title>chinaunix</title>
+	1 个或多个字符(贪婪匹配)	与上同理

续表

语　法	意　　义	说　　明
?	0 个或多个字符(贪婪匹配)	与上同理
{m,n}	对于前一个字符重复 m 到 n 次,{m}亦可	a{6}匹配 6 个 a,a{2,4}匹配 2～4 个 a
{m,n}?	对于前一个字符重复 m 到 n 次,并取尽可能少	aaaaaa 中 a{2,4}只会匹配 2 个
[]	表示一个字符集	[0—9]、[a—z]、[A—Z]、[^0]
|	或	A|B,或运算
(…)	匹配括号中的任意表达式	
(?#…)	注释,可忽略	

正则表达式转义字符如表 15.2 所示。

表 15.2　正则表达式转义字符

转义字符	意　　义
\A	只在字符串开始进行匹配
\Z	只在字符串结尾进行匹配
\b	匹配位于开始或结尾的空字符串
\B	匹配不位于开始或结尾的空字符串
\d	相当于[0—9]
\D	相当于[^0-9]
\s	匹配任意空白字符:[\t\n\r\v]
\S	匹配任意非空白字符:[^\t\n\r\v]
\w	匹配任意数字和字母:[a-zA-Z0-9]
\W	匹配任意非数字和字母:[^a-zA-Z0-9]

re 的主要功能函数包括 compile、match、search、split、findall(finditer)、sub(subn),具体用法如下。

(1) compile 函数格式如下:

```
re.compile(string[,flag])
```

flags 定义如下:

- re. I: 忽略大小写。
- re. L: 表示特殊字符集\w,\W,\b,\B,\s,\S,依赖于当前环境。
- re. M: 多行模式。
- re. S: 任意字符。
- re. U,表示特殊字符集\w,\W,\b,\B,\d,\D,\s,\S,依赖于 Unicode 字符属性数

据库。

【例 15-2】 compile 举例。

```
import re
pattern=re.compile(r'\d+')
print re.split(pattern,'one1two2three3four4')
```

运行结果：

```
['one', 'two', 'three', 'four', '']
```

（2）math 函数格式如下：

```
re.match(pattern, string[, flags])
```

【例 15-3】 match 举例。

```
import re
m=re.match('(\w+) (\w+)(?P<sign>.*)', 'hello world!')
print "m.string:", m.string
print "m.re:", m.re
print "m.pos:", m.pos
print "m.endpos:", m.endpos
print "m.lastindex:", m.lastindex
print "m.lastgroup:", m.lastgroup
print "m.group():", m.group()
print "m.group(1,2):", m.group(1, 2)
print "m.groups():", m.groups()
print "m.groupdict():", m.groupdict()
print "m.start(2):", m.start(2)
print "m.end(2):", m.end(2)
print "m.span(2):", m.span(2)
print "m.expand('\g \g\g'):", m.expand('\2 \1\3')
```

运行结果：

```
m.string: hello world!
m.re: <_sre.SRE_Pattern object at 0x012B22C8>
m.pos: 0
m.endpos: 12
m.lastindex: 3
m.lastgroup: sign
m.group(): hello world!
m.group(1,2): ('hello', 'world')
m.groups(): ('hello', 'world', '!')
m.groupdict(): {'sign': '!'}
m.start(2): 6
m.end(2): 11
```

```
m.span(2): (6, 11)
m.expand(r'\g \g\g'): world hello!
```

(3) search 函数格式如下：

```
re.search(pattern, string[, flags])
```

search 函数匹配并提取第一个符合规律的内容，返回一个正则表达式对象。

【例 15-4】 search 举例。

```
import re
s= 'asdfxxIxx123xxlovexxdfd'
a= re.search('xx(.* ?)xx123xx(.* ?)xx',s).group(1)
print a
b= re.search('xx(.* ?)xx123xx(.* ?)xx',s).group(2)
print b
c= re.findall('xx(.* ?)xx123xx(.* ?)xx',s)
print c
d= re.findall('xx(.* ?)xx123xx(.* ?)xx',s)
print d[0]
e= re.findall('xx(.* ?)xx123xx(.* ?)xx',s)
print e[0][1]
```

运行结果：

```
I
love
[('I', 'love')]
('I', 'love')
Love
```

(4) split 函数格式如下：

```
re.split(pattern, string[, maxsplit])
```

【例 15-5】 split 举例。

```
import re
pattern= re.compile(r'\d+ ')
print re.split(pattern,'one1two2three3four4')
```

运行结果：

```
['one', 'two', 'three', 'four', '']
```

(5) findall 函数格式如下：

```
re.findall(pattern, string[, flags])
```

findall 用于匹配所有符合规律的内容，返回包含结果的列表。

【例 15-6】 findall 举例。

```
import re
secret_code= 'hadkfalifexxIxxfasdjifja134xxlovexx23345sdfxxyouxx8dfse'
b= re.findall('xx.* xx',secret_code)                //贪心算法
print b
c= re.findall('xx.* ?xx',secret_code)               //非贪心算法
print c
d= re.findall('xx(.* ?)xx',secret_code)
print d
```

运行结果：

```
['xxIxxfasdjifja134xxlovexx23345sdfxxyouxx']
['xxIxx', 'xxlovexx', 'xxyouxx']
['I', 'love', 'you']
```

【解析】 Python 里数量词默认是贪婪的，即总是尝试匹配尽可能多的字符；而非贪婪则总是尝试匹配尽可能少的字符。* 属于贪匹配，而 * ?属于非贪匹配，例如，正则表达式 ab * 如果用于查找 abbbc，将找到 abbb。而如果使用非贪婪的数量词 ab * ?，将找到 a。

（6）finditer 函数格式如下：

```
re.finditer(pattern, string[, flags])
```

finditer 用于搜索 string，返回一个顺序访问每一个匹配结果（Match 对象）的迭代器。

【例 15-7】 finditer 举例。

```
import re
pattern= re.compile(r'\d+ ')
for m in re.finditer(pattern,'one1two2three3four4'):
    print m.group()
```

运行结果：

```
1 2 3 4
```

（7）sub 函数格式如下：

```
re.sub(pattern, repl, string[, count])
```

使用 repl 替换 string 中每一个匹配的子串后返回替换后的字符串。当 repl 是一个字符串时，可以使用\id 或\g 引用分组，但不能使用编号 0。当 repl 是一个方法时，这个方法应当只接受一个参数（Match 对象），并返回一个字符串用于替换（返回的字符串中不能再引用分组）。count 用于指定最多的替换次数，不指定时全部替换。

【例 15-8】 sub 举例。

```
import re
```

```
s='123abcssfasdfas123'
a=re.sub('123(.*?)123','123789123',s)
print a
b=re.sub('123(.*?)123','123%d123'%789,s)
print b
```

运行结果：

```
123789123
123789123
```

15.2.4　从网页上抓取特定信息

【例 15-9】　实现从网页上抓取图片。

```
#coding=utf-8
import re                                  #re 模块主要包含了正则表达式
import urllib                              #urllib 模块提供了读取 Web 页面数据的接口

def getHtml(url):                          #定义 getHtml()函数
    page=urllib.urlopen(url)               #urllib.urlopen()方法用于打开一个 URL 地址
    html=page.read()                       #read()方法用于读取 URL 上的数据
    return html

def getImg(html):
    reg=r'src="(.+?\.jpg)" pic_ext'           #正则表达式,得到图片地址
    imgre=re.compile(reg)                     #re.compile() 可以把正则表达式编译成一个正则表
                                               达式对象
    imglist=imgre.findall(html)               #re.findall() 方法读取 html 中包含 imgre 的数据

    x=0
    for imgurl in imglist:                     #通过 for 循环遍历筛选的图片地址并保存到本地
        urllib.urlretrieve(imgurl,'%s.jpg' %x)  #urllib.urlretrieve()直接将远程数据下载到本地
        x=x+1                                 #图片通过 x 依次递增来命名
url=raw_input("please input website:")
html=getHtml(url)
getImg(html)
```

运行结果：

```
C:\Python27\python.exe C:/Users/SELab/PycharmProjects/untitled/test.py
please input website: http://tieba.baidu.com/p/2460150866
```

【解析】　程序执行后，会将 http://tieba.baidu.com/p/2460150866 网址中的所有图片下载到与 py 文件相同的目录下，并以图片的次序命名为 0.jpg、1.jpg 等。

15.2.5 保存贴吧网页的小爬虫

【例 15-10】 实现保存贴吧网页的小爬虫。

【解答】

代码如下：

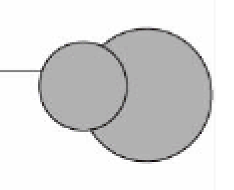
```
#-*-coding: utf-8-*-
import urllib2
def load_page(url):
    request=urllib2.Request(url)
    response=urllib2.urlopen(request)
    the_page=response.read()
    return the_page
def write_to_file(filename,txt):
    '''
    将 txt 文本存入到 html 文件中
    '''
    print filename + ' is saved ... '
    f=open(filename,'w')
    f.write(txt)
    f.close()
def tieba_spider(url,begin_page,end_page):
    for i in range(begin_page,end_page+1):
        pn=50*(i-1)        //贴吧每页显示 50 条数据
        my_url=url + str(pn)
        my_page=load_page(my_url)
        filename=str(i) + '.html'
        write_to_file(filename,my_page)
#main
if __name__=="__main__":
    url=raw_input('please input your url:')
    begin_page=int(raw_input("please input the begin page:"))
    end_page=int(raw_input("please input the end page:"))
    tieba_spider(url,begin_page,end_page)
```

运行结果：

```
please input your url:
http://tieba.baidu.com/f?ie=utf-8&kw=%E8%A5%BF%E5%AE%89%E9%82%AE%E7%94%B5%E5%A4%A7%E5%AD%A6&fr
=search
please input the begin page:1
please input the end page:2
1.html is saved ...
2.html is saved ...
```

【解析】 以上输入了西安邮电大学贴吧的网址以及起始、终止的页码，在 py 文件相同的目录下可以得到 1.html 和 2.html 文件，内容为西安邮电大学贴吧的网页内容。

第 16 章



软件测试框架

16.1 本章要求

- 了解软件测试的基本概念和测试分类。
- 了解 Python 的测试框架。
- 掌握单元测试工具 PyUnit。
- 掌握 GUI 测试工具 pywinauto。
- 掌握 Web 测试工具 Selenium。
- 掌握性能测试工具 Pylot。

16.2 本章知识重点

16.2.1 Python 与软件测试

1. 软件测试

软件测试是控制软件质量的重要手段和关键活动，其主要意义如下：

(1) 测试并不仅仅是为了找出错误，通过分析错误产生的原因和错误的发生趋势，可以帮助项目管理者发现当前软件开发过程中的缺陷，以便及时改进。

(2) 测试可以帮助测试人员设计出有针对性的测试方法，改善测试的效率和有效性。

(3) 没有发现错误的测试也是有价值的，完整的测试是评定软件质量的一种方法。

软件测试的分类方法很多，具体如图 16.1 所示。

自动化测试就是通过测试工具实现手工测试无法实现的功能，减轻手工测试的工作量，提高了测试效率。以下场合往往会优先考虑使用自动化测试。

(1) 软件需求变动不频繁。

当软件需求变动过于频繁时，势必多次更新测试用例以及测试脚本，增大维护脚本的成本。一般情况下，对于需求中相对稳定的模块进行自动化测试，对于变动较大的模块采用手工测试。

(2) 项目周期足够长。

由于自动化测试需求的确定、自动化测试框架的设计、测试脚本的编写与调试需要相

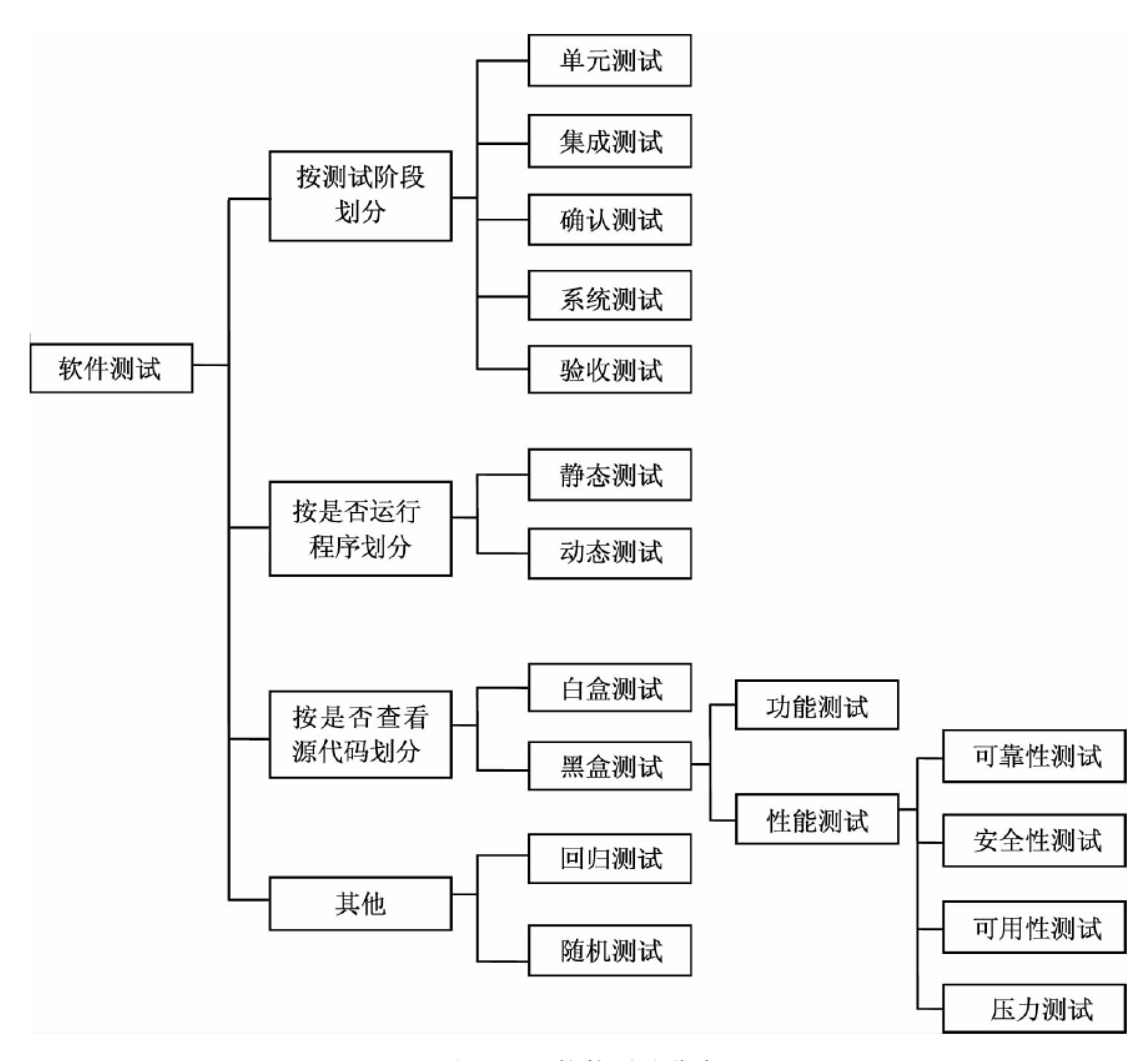


图 16.1 软件测试分类

当长的时间来完成，因此只在项目周期足够长的情况下采用。

(3) 自动化测试脚本可重复使用，手工测试完成难度较大的场合。

性能测试、压力测试、负载测试等需要模拟大量并发用户时，自动化测试脚本可重复使用。许多与时序、死锁、资源冲突、多线程等有关的软件缺陷很难通过手工测试的场合，也适合采用自动化测试。

(4) 回归测试。

回归测试在软件每次有新版本时都必须执行，也就是在软件的生命周期中会被反复执行的测试，因此这类测试很适合采用自动化测试。

2. Python 测试框架

Python 的测试框架大致分为如下几方面：

1) PyUnit 单元测试

单元测试是由开发人员(而不是测试人员)完成的测试，用于保证一个程序基本单元的正确性。单元测试框架代替开发人员完成了一些调用、I/O 等与单元测试无直接关系的支撑代码，让开发人员可以专注于测试用例的编写，简化单元测试工作。

Python 自带 unittest 模块实现单元测试，与单元测试 JUnit 类似。

2）Windows GUI 测试

GUI(Graphical User Interface，图形用户界面)是计算机软件与用户进行交互的主要方式。GUI 测试是指对 GUI 开发环境的可复用的构件进行操作的正误判断。

Python 提供 pywinauto 开源框架进行测试，与 QTP 测试工具功能类似。

3）Web 自动化负载测试

目前的自动化负载测试解决方案几乎都采用“录制-回放”的技术。通过捕获用户每一步操作，如界面的像素坐标或对象(窗口、按钮、滚动条等)的位置以及状态或属性的变化，用脚本语言记录下来。回放时，将脚本语言转换为屏幕操作，对比被测系统的输出记录与预先设计的标准记录之间的关系。

Python 进行 Web 自动化测试的工具有很多，如 Selenium、RF 和 twill 等，其中，较为常用的是 Selenium。

4）压力性能测试

软件压力测试是指被测应用程序能够承受的负荷，具体是指同时能够承受的用户访问量(容量)，即最多支持有多少用户同时访问某个功能。

Python 提供了 Pylot 工具实现压力性能测试。

16.2.2 用 PyUnit 进行单元测试

Python 单元测试框架(Python Unit testing framework，简称 PyUnit)是 Kent Beck 和 Erich Gamma 设计的 JUnit 的 Python 版本。

PyUnit 操作步骤如下：

步骤 1：执行 import unittest。

步骤 2：定义一个继承自 unittest. TestCase 的测试用例类。

步骤 3：定义 setUp 和 tearDown，在每个测试用例前后做一些辅助工作。

步骤 4：定义测试用例，名字以 test 开头。PyUnit 通过调用 assertEqual、assertRaises 等断言方法判断程序执行结果和预期值是否相符。

步骤 5：调用 unittest. main()启动测试。

步骤 6：如果测试未通过，会输出相应的错误提示。

【例 16-1】 PyUnit 测试举例。

```
import unittest
class Person:                                  #将要被测试的类
    def age(self):
        return 34
    def name(self):
        return 'bob'
class PersonTestCase(unittest.TestCase):       #测试用例类继承 unittest.TestCase
    def setUp(self):
```

```
            self.man= Person()
            print 'set up now'
        def test1(self):                                #测试用例
            self.assertEqual(self.man.age(), 34)      #动态测试方法 assertEqual()
        def test2(self):
            self.assertEqual(self.man.name(), 'bob')
        def test3(self):
            self.assertEqual(223,23)
if __name__=='__main__':
    unittest.main()
```

程序运行结果如图 16.2 所示。

```
set up now
set up now
set up now
..F
======================================================================
FAIL: test3 (__main__.PersonTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:/Users/Administrator/PycharmProjects/test1/hello world.py", line 17, in test3
    self.assertEqual(223,23)
AssertionError: 223 != 23

----------------------------------------------------------------------
Ran 3 tests in 0.000s

FAILED (failures=1)
```

图 16.2　PyUnit 运行结果

【解析】 test1、test2 运行正确，而 test3 测试失败，分析 self.assertEqual(223,23)代码可知错误原因。

16.2.3　用 pywinauto 进行 GUI 测试

pywinauto 用于测试 Windows 控件的一系列动作，如指定窗口、鼠标或键盘操作、获得控件属性等。pywinauto 的官网网址是：http://pypi.python.org/pypi/pywinauto/0.4.0

pywinanto 的安装步骤如下：

步骤 1：下载 pywinauto，其下载网址是 https://sourceforge.net/projects/pywinauto/files/，如图 16.3 所示。

步骤 2：将下载的 pywinauto-0.4.0.zip 文件解压缩到 C:\pywinauto-0.4.0，在命令行窗口中进入 pywinauto-0.4.0 目录，执行如下安装命令：python setup.py install，如图 16.4 所示。

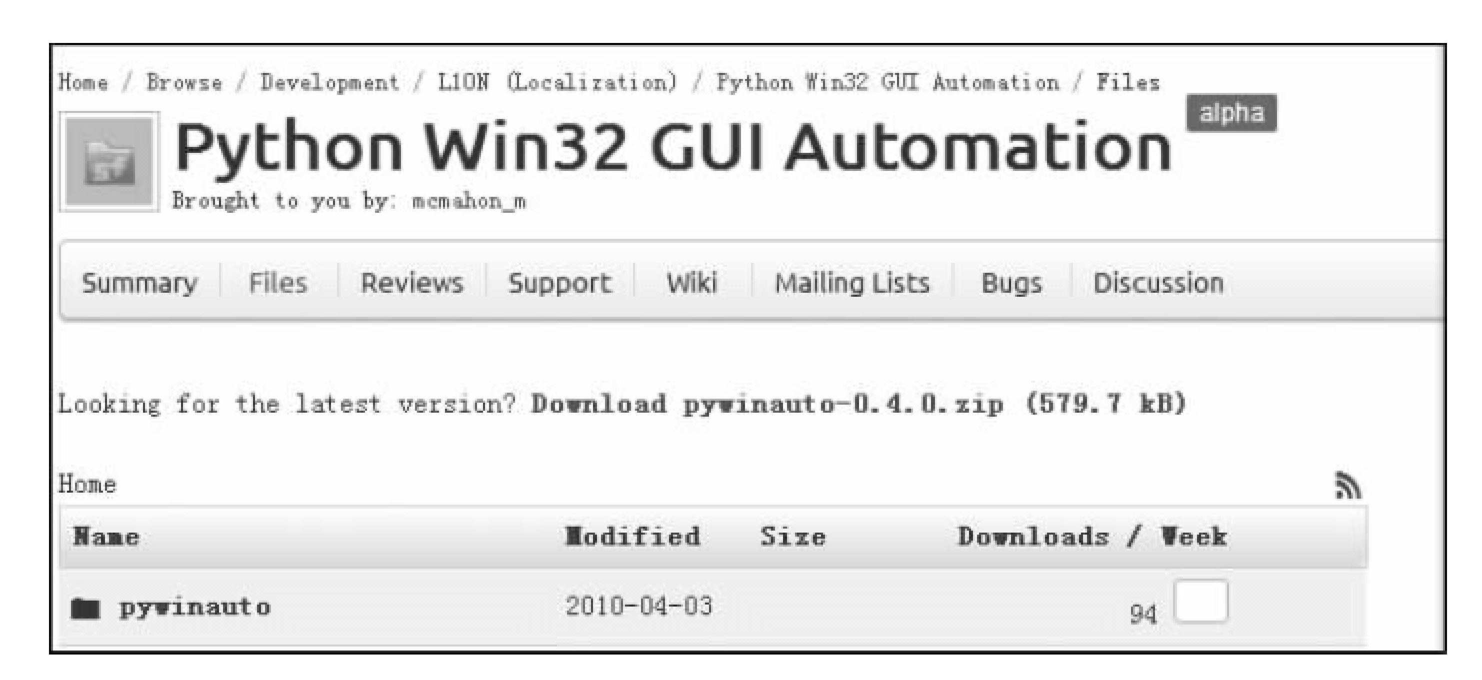


图 16.3　pywinauto 下载网页

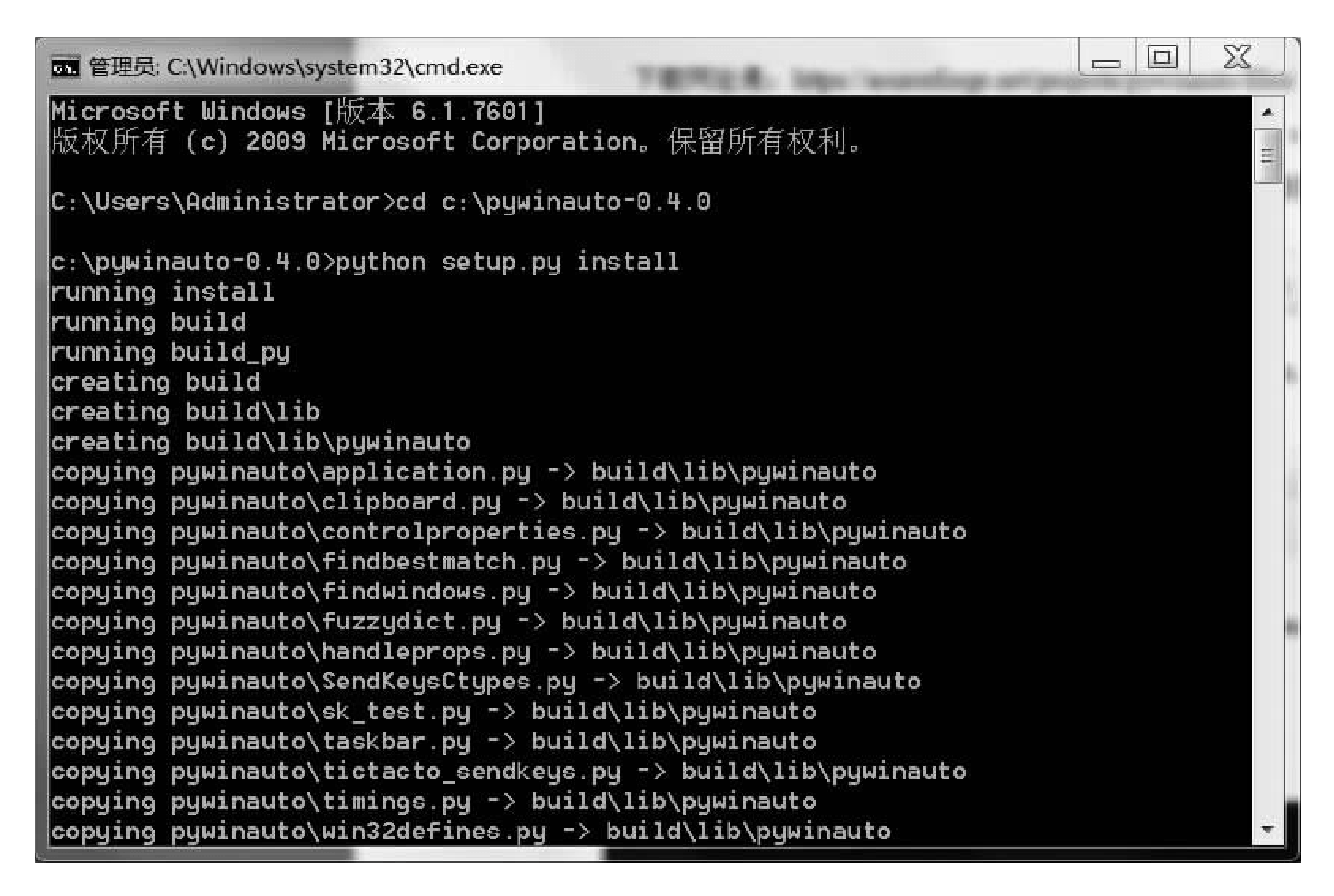


图 16.4　pywinauto 安装

【例 16-2】　pywinauto 举例。

```
from pywinauto import Application                    #引入 pywinauto 模块
import time
app= Application.start("notepad")                    #调用 notepad
app.__setattr__("name","notepad")
time.sleep(2)
app.Notepad.edit.TypeKeys('hello!')
i=0
while i<=10:
    app.Notepad.edit.TypeKeys('test')
    i+=1
time.sleep(2)
app.Dialog.Button1.Click()
```

```
time.sleep(1)
app.Notepad.Close()
```

程序运行结果如图 16.5 所示。



图 16.5 pywinauto 运行结果

16.2.4 用 Selenium 进行 Web 测试

Selenium 是开源的自动化软件测试工具，用于测试 Web 应用程序在不同的浏览器和操作系统中的运行情况。Selenium 工具集包括 Selenimu IDE、Selenium Remote Control（Selenium RC）、Selenium WebDriver 和 Selenium Grid，如表 16.1 所示。

表 16.1 Selenium 工具集简介

工　具	描　述
Selenium IDE	Selenium 集成开发环境（IDE）是一个 Firefox 插件，可以让测试人员跟踪，需要测试的工作流程，以记录其行为
Selenium RC	Selenium 远程控制（RC）为旗舰测试框架，它允许多个简单的浏览器动作和线性执行。它能使用编程语言，如 Java、C＃、PHP、Python、Ruby 和 Perl 的强大功能来创建更复杂的测试
Selenium WebDriver	Selenium WebDriver 前身是 Selenium RC，直接发送命令给浏览器，并检索结果
Selenium Grid	Selenium Grid 是运行在不同的机器、不同的浏览器，同时可以最小化执行时间的并行测试工具

本书重点介绍 Selenium IDE，它是 Firefox 浏览器的插件，能够录制并回放用户在 Firefox 中操作的行为，并存储为 HTML、Java、C＃、Python 等脚本语言。

1. 安装 Selenium IDE

打开 FireFox 浏览器，选择“工具”→“附件组件”→“获取添加组件”菜单命令，找到插件安装页面，在搜索栏输入 selenium ide 进行搜索，选择 Selenium IDE 进行安装，如图 16.6 所示。



图 16.6　Selenium IDE 的安装

Selenium IDE 安装成功后重启 Firefox，在“工具”菜单栏下可以看到 Selenium IDE 菜单项，如图 16.7 所示。

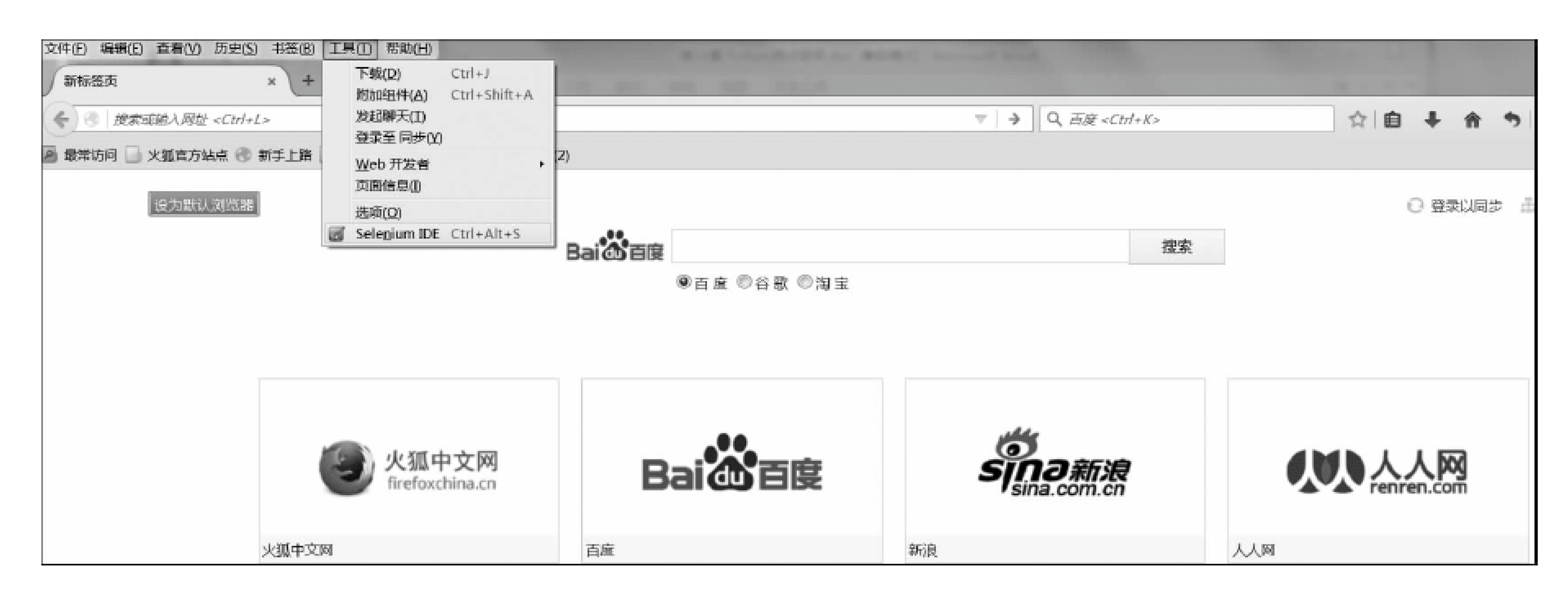


图 16.7　安装 Selenium IDE 后的“工具”菜单

打开 Selenium IDE，进入 Selenium IDE 主页面，如图 16.8 所示。

2. Selenium IDE 测试

1）录制功能

步骤 1：启动 Firefox 浏览器，输入网址 www.baidu.com。

步骤 2：从工具菜单中打开 Selenium IDE，Base URL 中默认为 www.baidu.com，如图 16.9 所示。

步骤 3：在 Firefox 中进行操作，在百度中输入 Selenium IDE 进行操作，操作行为会被 Selenium IDE 转化为相应的命令，出现在 Table 选项卡中，每一条都由 3 个部分组成：

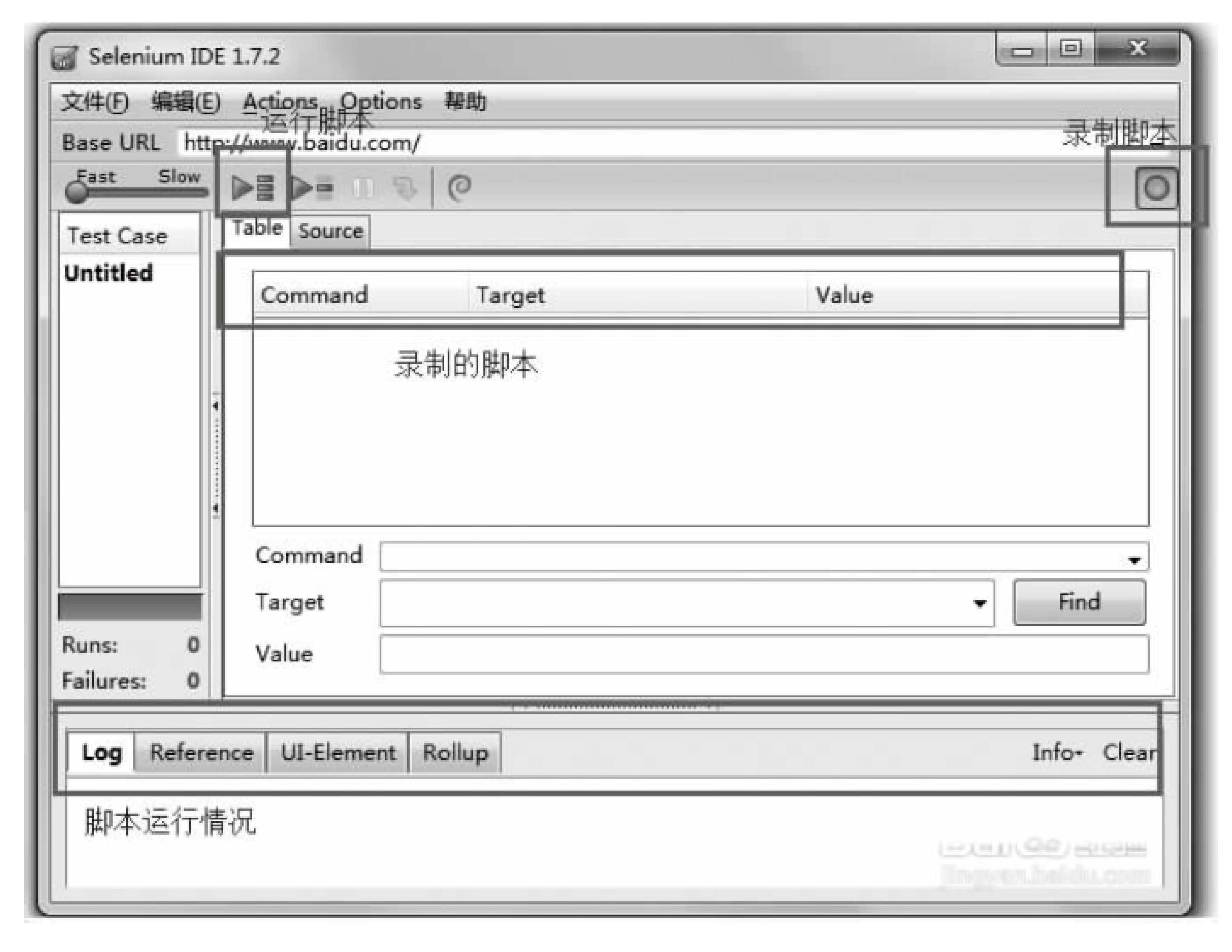


图 16.8 Selenium IDE 的安装

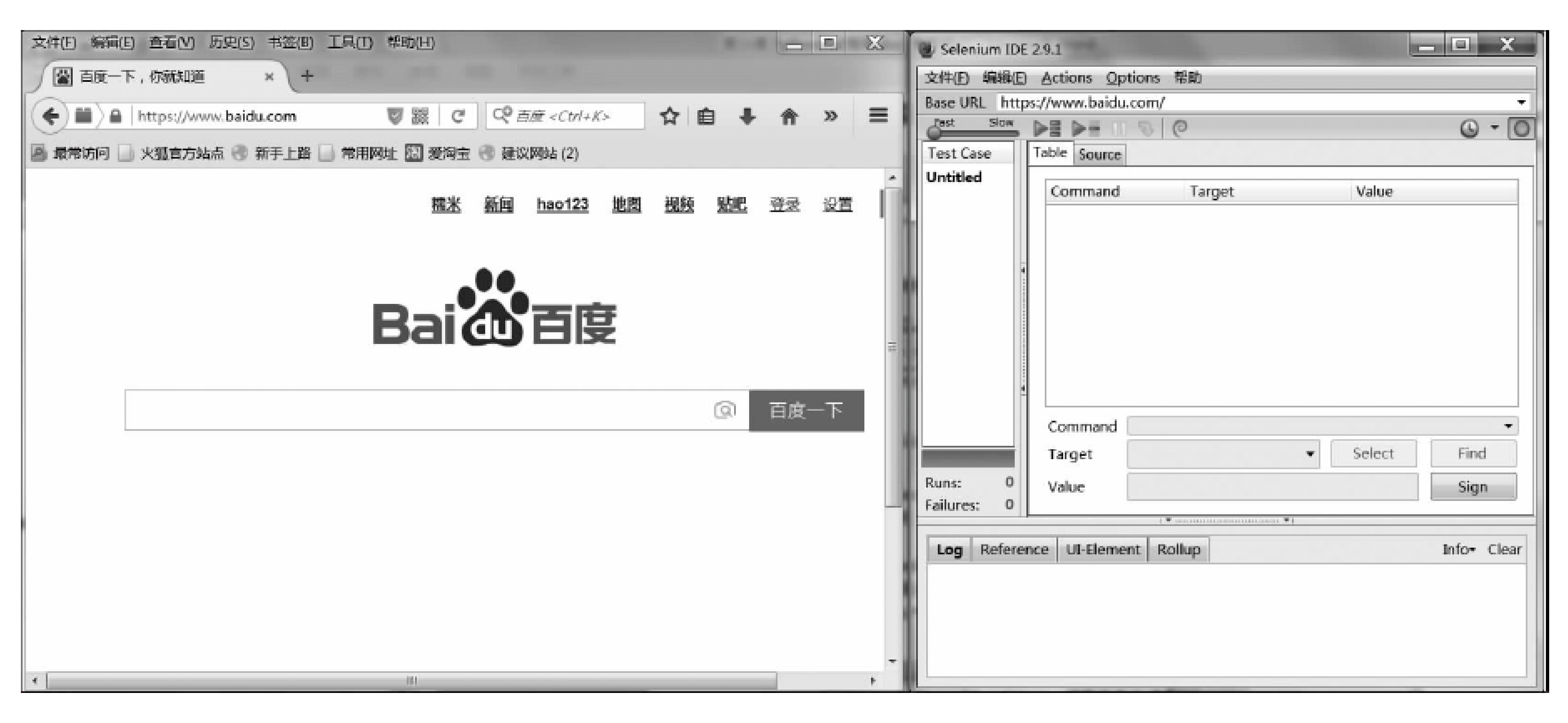


图 16.9 打开 Selenium IDE

Command(命令,如单击(click))、Target(目标,即命令的作用对象,如单击选中的按钮)、Value(值,如输入框中的文本字符串),如图 16.10 所示。

步骤 4: 在 Selenium IDE 主页面单击 Base URL 输入框右下方的红色按钮,停止录制。随后会发现 Selenium IDE 的 Source 框中显示类似 html 的脚本,这是录制过程中生成的测试脚本,用于回放。

录制脚本默认生成 HTML 格式,也可选择 Options→Format 菜单生成其他语言的脚本,如 Java、C#、Python 等。录制的脚本要通过"文件"菜单来保存,如图 16.11 所示。

2) 回放功能

在 Selenium IDE 主页面单击运行脚本按钮开始回放后,在 Firefox 浏览器中看到

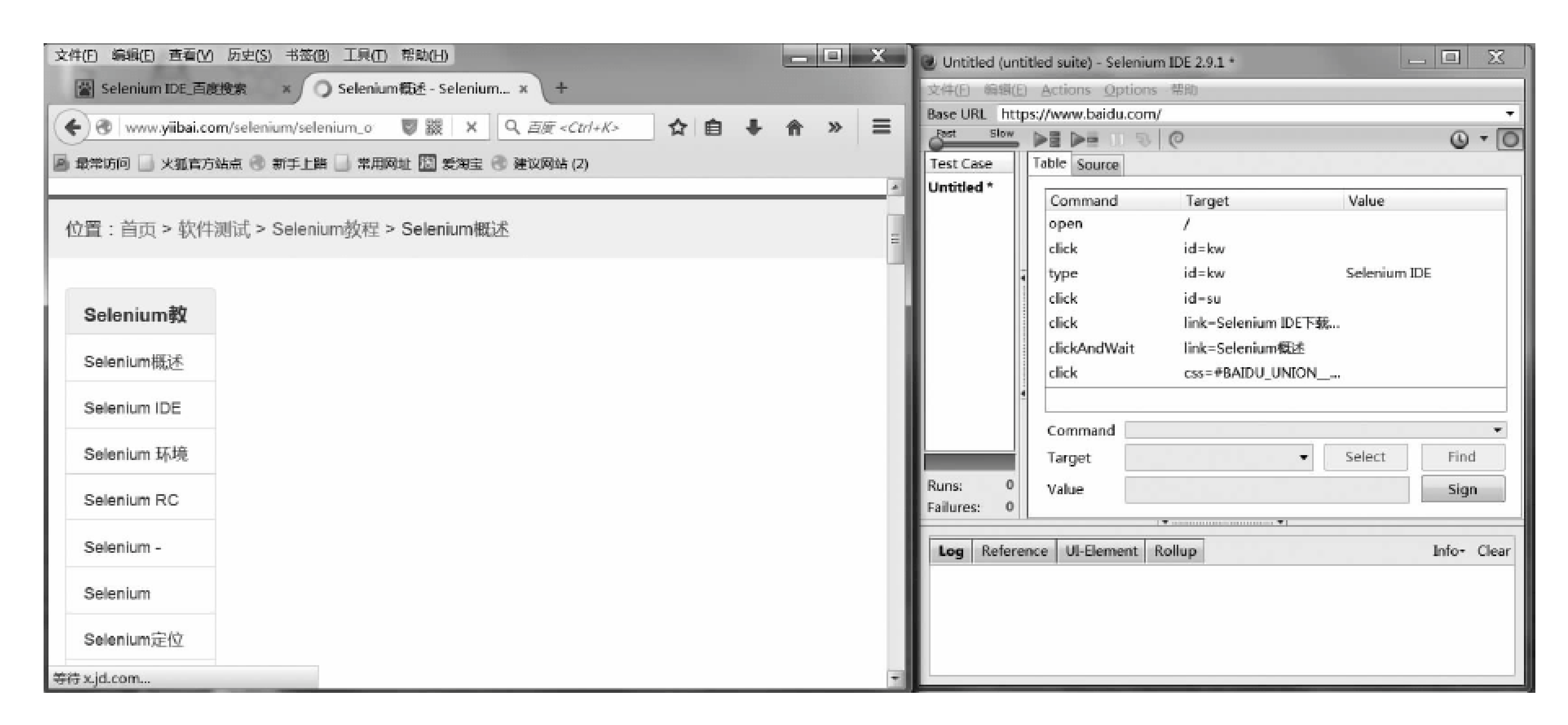


图 16.10 录制动作

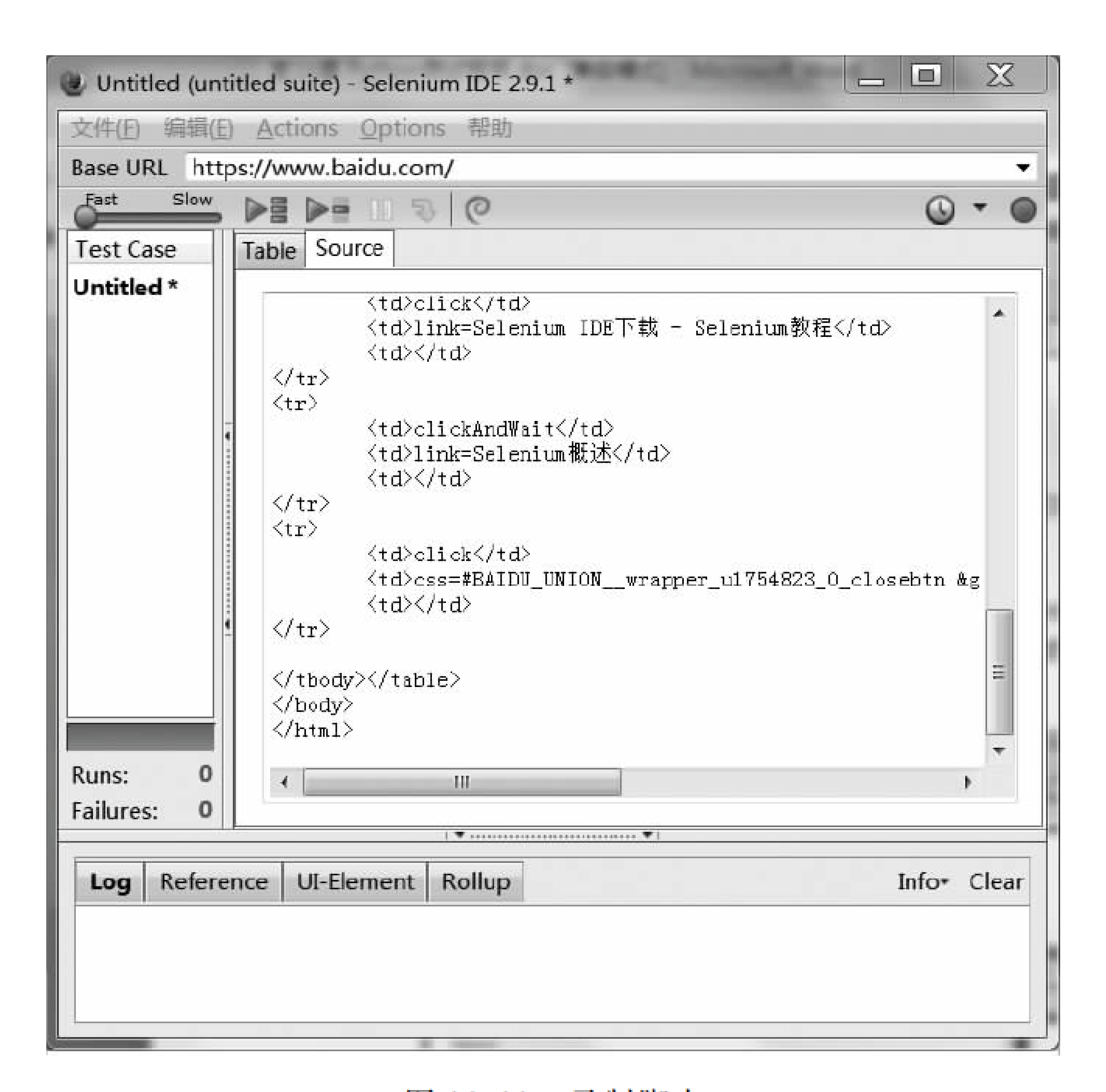


图 16.11 录制脚本

IDE 在自动回放先前录制的动作，如图 16.12 所示。

回放的过程中往往会出现一些问题，可以通过添加相关命令修改脚本，如图 16.13 所示。相关资料可参考 http://www.ltesting.net/ceshi/open/kygncsgj/selenium/。

16.2.5 用 Pylot 进行性能测试

Pylot 是 Python 编写的用以测试 Web 性能和扩展性的工具，进行 HTTP 负载测试。



图 16.12 自动回放

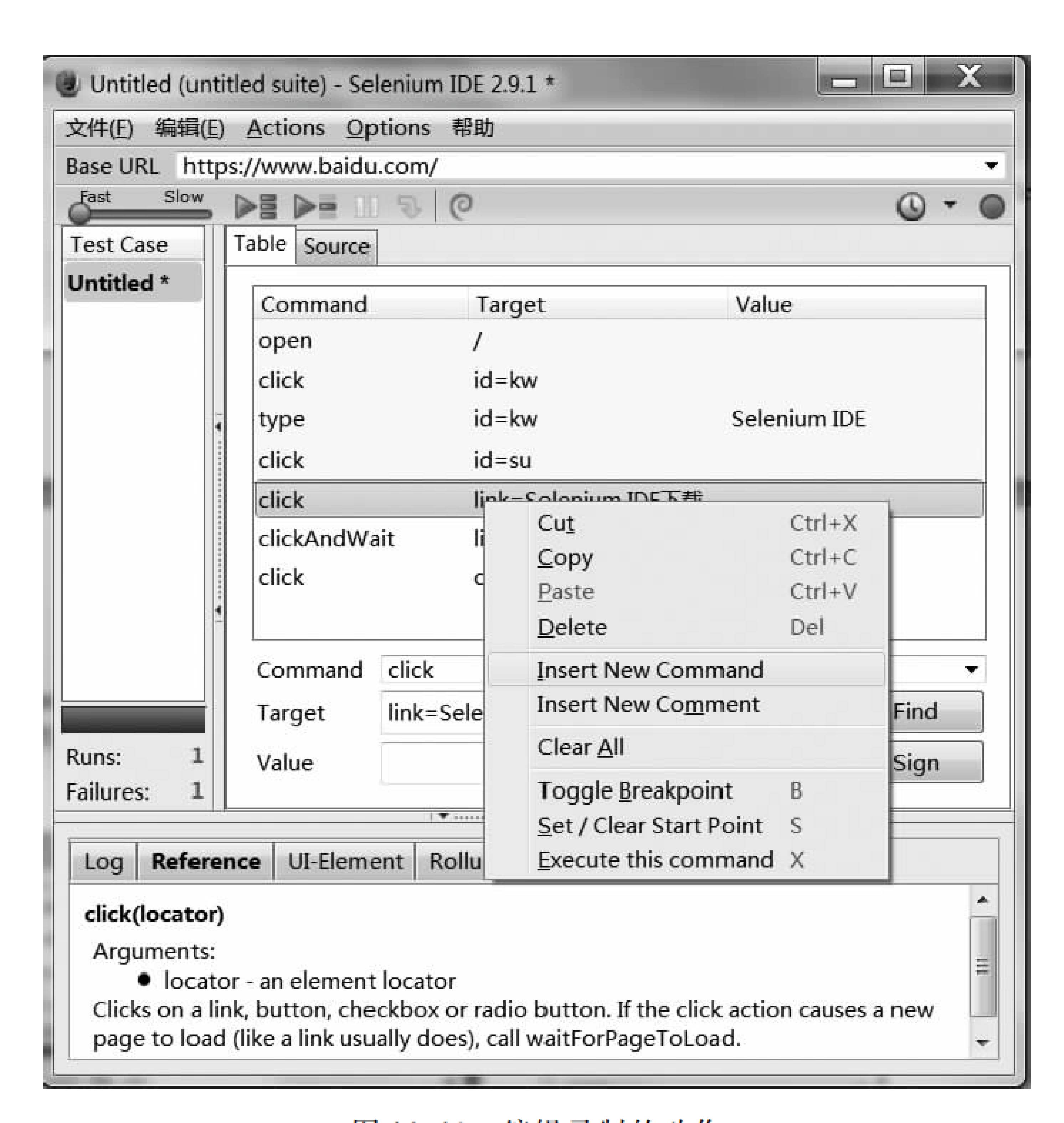


图 16.13 编辑录制的动作

Pylot 下载网址为 http://www.pylot.org/，如图 16.14 所示。

Pylot 的具体使用步骤如下：

步骤 1：下载完 Pylot 之后，解压到一个目录下，例如 C:\pylot_1.26，无须安装。

步骤 2：配置 testcases.xml。在 pylot_1.26 文件夹里，会看到一个 testcases.xml 的



图 16.14 Pylot 下载网址

文件，用记事本打开，把需要测试的网页地址添加进去。

```
<testcases>
<!-- SAMPLE TEST CASE-->
<case>
<url>http://www.example.com/</url>
</case>
<!-- SAMPLE TEST CASE-->
<!--
<case>
<url>http://search.yahooapis.com/WebSearchService/V1/webSearch</url>
<method>POST</method>
<body><![CDATA[appid=YahooDemo&query=pylot]]></body>
<add_header>Content-type: application/x-www-form-urlencoded</add_header>
</case>
-->
</testcases>
```

上面的代码中，把 http://www.example.com/改为要测试的网址，然后保存文件。

步骤 3：打开命令行窗口(选择“开始”→“运行”，输入 cmd，单击“确定”按钮)，进入 Pylot 的目录，在命令行输入如下命令：python run.py -a 20 -d 10，如图 16.15 所示。

上面的命令中，-a 表示并发 20 个客户端连接，-d 表示持续运行时间为 10s。

步骤 4：测试结束后，会在 Pylot 的文件目录里生成一个 results 文件夹，还生成一个 results.html 文件，用于记录详细的测试数据，如图 16.16 所示。

```
C:\WINDOWS\system32\cmd.exe

C:\pylot_1.26>python run.py -a 20 -d 10

-------------------------------------------------
Test parameters:
  number of agents:          20
  test duration in seconds:  10
  rampup in seconds:         0
  interval in milliseconds:  0
  test case xml:             testcases.xml
  log messages:              False

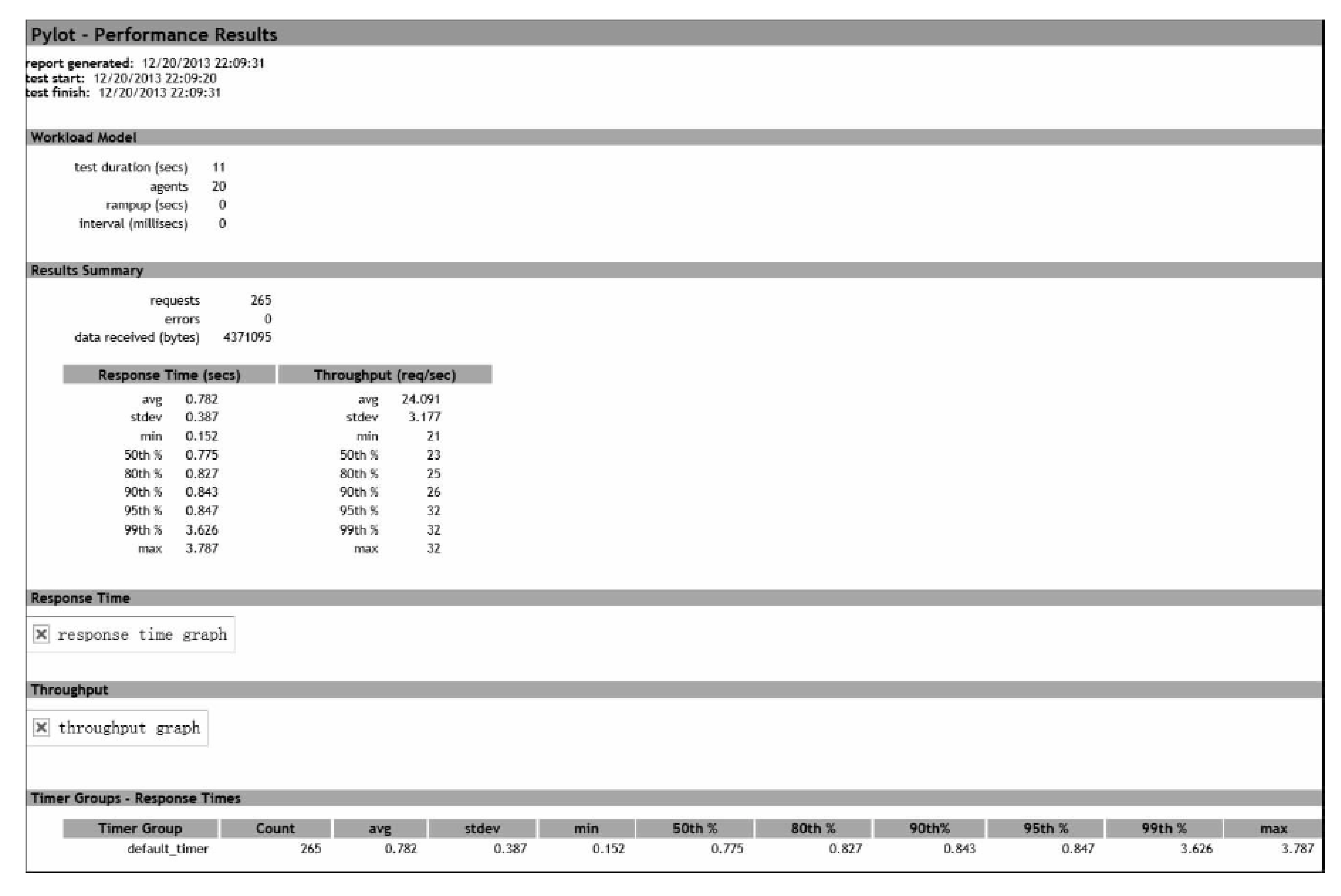

Started agent 20

All agents running...


[################100%##################]  10s/10s

Requests:  246
Errors: 0
Avg Response Time:  0.766
Avg Throughput:  24.33
Current Throughput:  20
Bytes Received:  4057680
```

图 16.15　Pylot 操作截图



Pylot - Performance Results

report generated: 12/20/2013 22:09:31
test start: 12/20/2013 22:09:20
test finish: 12/20/2013 22:09:31

Workload Model

test duration (secs)	11
agents	20
rampup (secs)	0
interval (millisecs)	0

Results Summary

requests	265
errors	0
data received (bytes)	4371095

Response Time (secs)		Throughput (req/sec)	
avg	0.782	avg	24.091
stdev	0.387	stdev	3.177
min	0.152	min	21
50th %	0.775	50th %	23
80th %	0.827	80th %	25
90th %	0.843	90th %	26
95th %	0.847	95th %	32
99th %	3.626	99th %	32
max	3.787	max	32

Response Time

response time graph

Throughput

throughput graph

Timer Groups - Response Times

Timer Group	Count	avg	stdev	min	50th %	80th %	90th%	95th %	99th %	max
default_timer	265	0.782	0.387	0.152	0.775	0.827	0.843	0.847	3.626	3.787

图 16.16　Pylot 运行结果

注意：Pylot 1.26 只支持 Python 2.5 和 Python 2.6。

16.3　习题与解答

16.3.1　习题

1. 简答题

1. 自动化测试相对于手工测试有什么好处？
2. 白盒测试是什么？
3. 黑盒测试是什么？
4. 采用 Pylot 对百度主页进行压力测试，主机数为 10 台，持续测试时间为 3s，将测试结果用 NumPy 和 Matplotlib 进行分析和图示化。

16.3.2　习题参考答案

1. 自动化测试相对于手工测试有什么好处？

【解答】 略

2. 白盒测试是什么？

【解答】 白盒测试是把测试对象看作一个打开的盒子，允许测试人员利用程序内部的逻辑结构及有关信息，设计或选择测试用例，通过在不同点检查程序状态，确定实际状态是否与预期的状态一致。

白盒测试分为静态测试和动态测试。静态白盒测试是在不执行程序的条件下审查软件设计、体系结构和代码，找出软件缺陷的过程。动态白盒测试也称结构化测试，通过运行程序，分析代码的内部结构和设计。

3. 黑盒测试是什么？

【解答】 黑盒测试也称功能测试，着眼于程序外部结构，不考虑内部逻辑结构，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，检查程序功能是否符合需求规格说明书的要求，程序是否能适当地接收输入数据而产生正确的输出信息。

4. 采用 Pylot 对百度主页进行压力测试，主机数为 10 台，持续测试时间为 3s，将测试结果用 NumPy 和 Matplotlib 进行分析和图示化。

【解答】 在测试前，需先修改 xml 文件中的网址，进入 Pylot_1.26 的文件夹中，以文本形式打开 testcases.xml 文件，将 xml 文件中的网址替换为 baidu 网站，如图 16.17 所示。

题意为采用 Pylot 并发 10 台主机，持续测试时间为 3s：

```
python run.py- a 10- d 3
```

其中，a 代表并发的主机数，d 表示测试时间。

测试执行过程如图 16.18 所示。

当测试完成时，会返回如图 16.19 所示的信息。此信息为生成测试报表的位置，只有在此信息出来时，整个测试过程才算正式结束，若提前关闭，则无法生成测试报表。

```
<testcases>
  <!-- SAMPLE TEST CASE -->
  <case>
    <url>http://www.baidu.com</url>
  </case>

  <!-- SAMPLE TEST CASE -->
  <!--
  <case>
    <url>http://search.yahooapis.com/WebSearchService/V1/webSearch</url>
    <method>POST</method>
    <body><![CDATA[appid=YahooDemo&query=pylot]]></body>
    <add_header>Content-type: application/x-www-form-urlencoded</add_header>

  </case>
  -->
</testcases>
```

图 16.17 修改需要测试的网站

```
C:\pylot_1.26>python run.py -a 10 -d 3

-------------------------------------------------
Test parameters:
  number of agents:          10
  test duration in seconds:  3
  rampup in seconds:         0
  interval in milliseconds:  0
  test case xml:             testcases.xml
  log messages:              False


Started agent 10

All agents running...


[##################100%###################]  3s/3s

Requests:  133
Errors: 0
Avg Response Time:  0.097
Avg Throughput:  43.06
Current Throughput:  02
Bytes Received:  8102271

-------------------------------------------------
```

图 16.18 测试过程

```
Generating Results...
Generating Graphs...

Done generating results. You can view your test at:
results/results_2016.05.29_19.29.11/results.html

Done.
```

图 16.19 测试完成信息

在报表生成后，在 Pylot_1.26 文件夹下的 result 文件夹中会有测试完成后所生成的数据图表，如图 16.20 及图 16.21 所示。

正确安装了 NumPy 和 Matplotlib 后，参照 11.2.1 节和 11.2.2 节的相关内容操作，会显示响应时间及吞吐量的实时变化图，如图 16.22 及图 16.23 所示。

Pylot - Performance Results

report generated: 05/04/2016 17:27:30
test start: 05/04/2016 17:22:20
test finish: 05/04/2016 17:22:35

Workload Model

test duration (secs)	15
agents	10
rampup (secs)	0
interval (millisecs)	0

Results Summary

requests	131
errors	0
data received (bytes)	13019286

Response Time (secs)		Throughput (req/sec)	
avg	0.690	avg	9.357
stdev	1.258	stdev	7.459
min	0.079	min	1
50th %	0.110	50th %	10
80th %	0.971	80th %	16
90th %	1.903	90th %	20
95th %	3.787	95th %	24
99th %	7.201	99th %	24
max	7.421	max	24

图 16.20　测试图表一

Timer Groups - Response Times

Timer Group	Count	avg	stdev	min	50th %	80th %	90th%	95th %	99th %	max
default_timer	131	0.690	1.258	0.079	0.110	0.971	1.903	3.787	7.201	7.421

Agent Details

Agent	Start Time	Requests	Errors	Bytes Received	Avg Response Time (secs)	Avg Connect Time (secs)
1	17:22:20	20	0	1987672	0.532	0.211
2	17:22:20	9	0	894667	1.146	0.246
3	17:22:20	12	0	1192236	0.399	0.104
4	17:22:20	17	0	1689325	0.706	0.157
5	17:22:20	6	0	596510	0.422	0.099
6	17:22:20	6	0	596089	0.415	0.352
7	17:22:20	15	0	1490817	0.709	0.261
8	17:22:20	19	0	1888233	0.616	0.226
9	17:22:20	8	0	795463	1.812	0.314
10	17:22:20	19	0	1888274	0.569	0.220

Fastest Responding URLs

Request URL	Avg Response Time (secs)
http://www.baidu.com	0.690

Slowest Responding URLs

Request URL	Avg Response Time (secs)
http://www.baidu.com	0.690

图 16.21　测试图表二

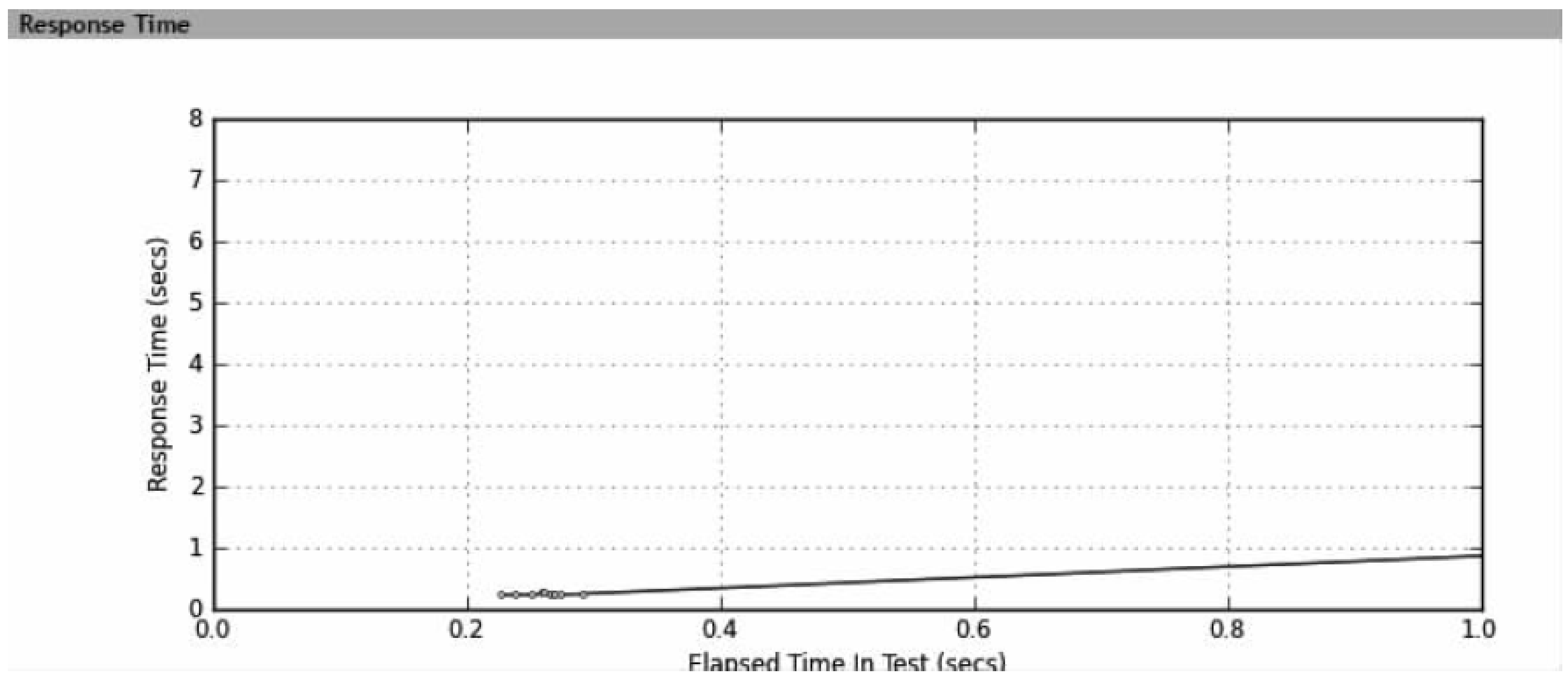


图 16.22　响应时间图

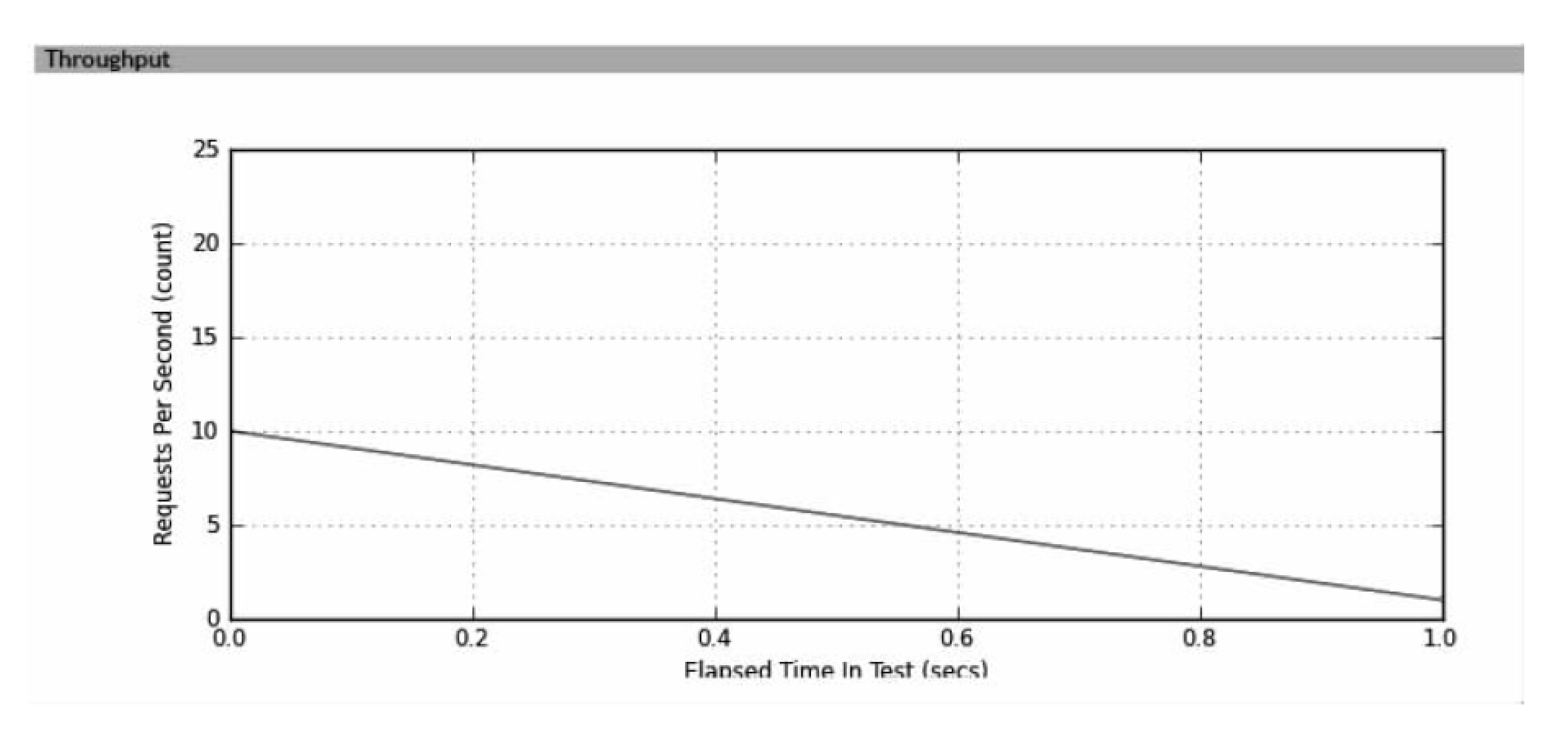
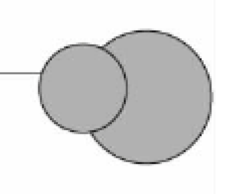

Throughput
Requests Per Second (count)
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
Elapsed Time In Test (secs)

图 16.23　吞吐量图

第 17 章



Web开发框架

17.1 本章要求

- 了解 MVC 设计模式。
- 掌握 web2py 框架。
- 掌握 Django 框架。

17.2 本章知识重点

17.2.1 MVC 设计模式

MVC 模式是当前较为流行的三层模式，MVC 是 Model、View 和 Control 3 个单词的缩写，是指该模式由模型、视图和控制器 3 部分组成，其框架如图 17.1 所示。

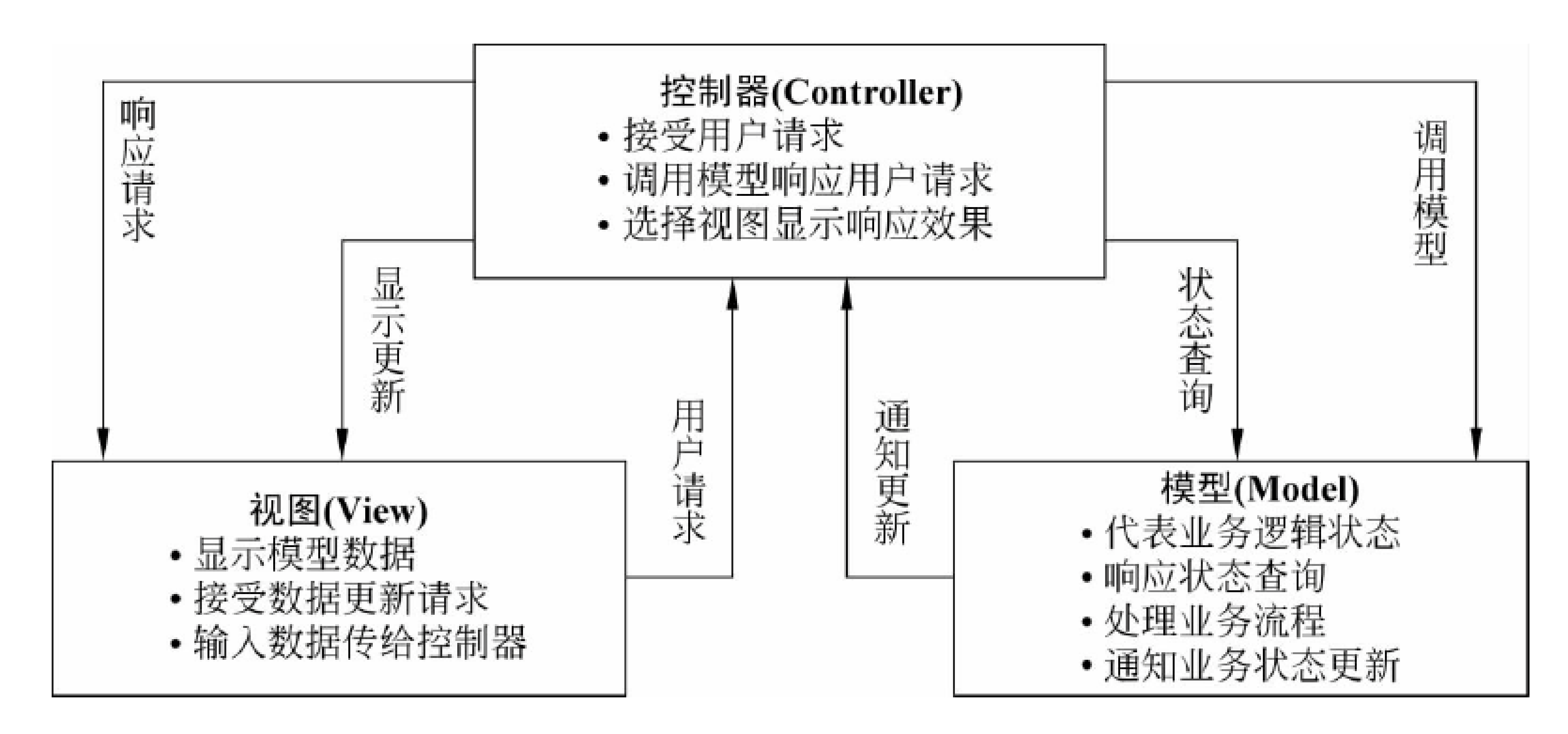


图 17.1　MVC 设计模式

- 模型。模型封装了数据和基于这些数据的操作，是系统的业务逻辑部分。
- 视图。视图提供了对模型的显示，是系统的用户界面部分。
- 控制器。控制器相当于模型和视图的一个中介，控制器接受所有视图提交的请求，根据请求内容的不同而转发到不同的模型中，并将最终结果回传给视图向用户显示，控制器是模式中最重要的部分。

17.2.2　web2py 框架

web2py 于 2007 年发布，作为免费、开源的 Web 开发框架，使得 Web 开发简单、快捷。web2py 下载网址为 http://www.web2py.com，如图 17.2 所示。



图 17.2　web2py 下载网址

web2py 下载页面如图 17.3 所示。

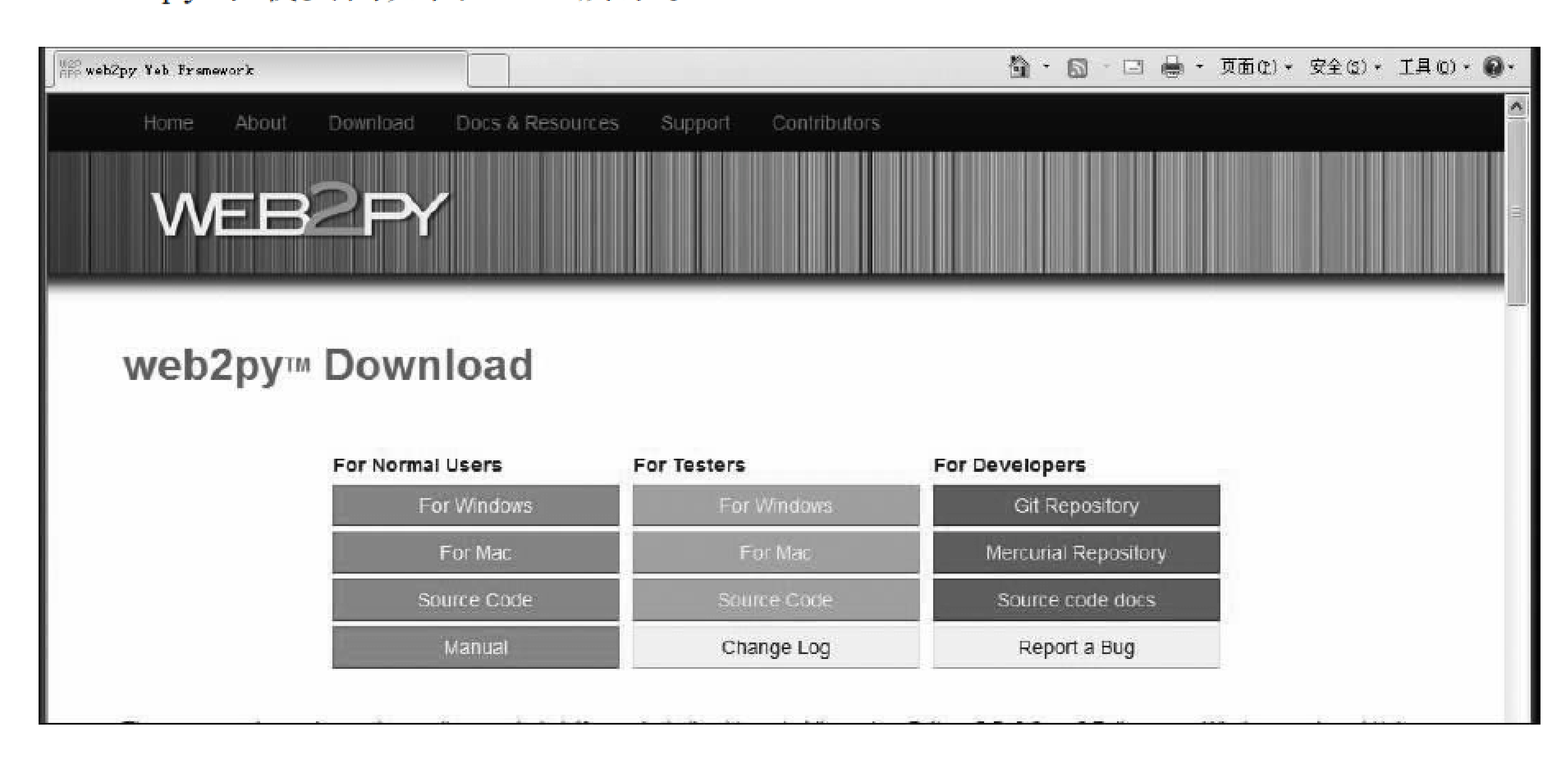


图 17.3　web2py 下载页面

单击 For Normal Users→For Windows，下载 web2py_win，解压缩后，双击 web2py.exe 文件进行安装，如图 17.4 所示。

第一次使用时直接输入预设的密码即可，系统会自动保存，并默认为以后登录使用的密码。输入密码后，出现 web2py 的欢迎界面，如图 17.5 所示。

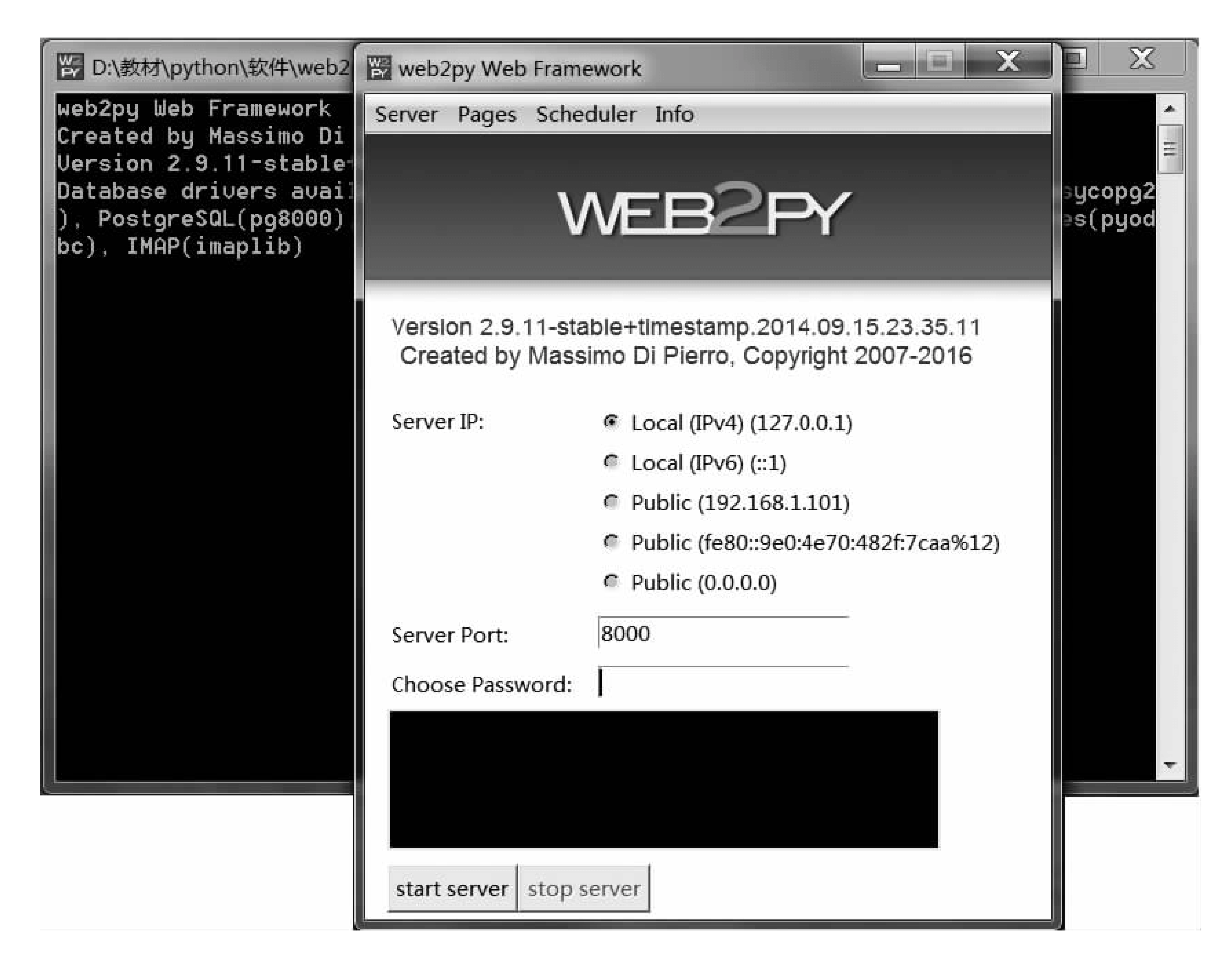


图 17.4　web2py 安装步骤

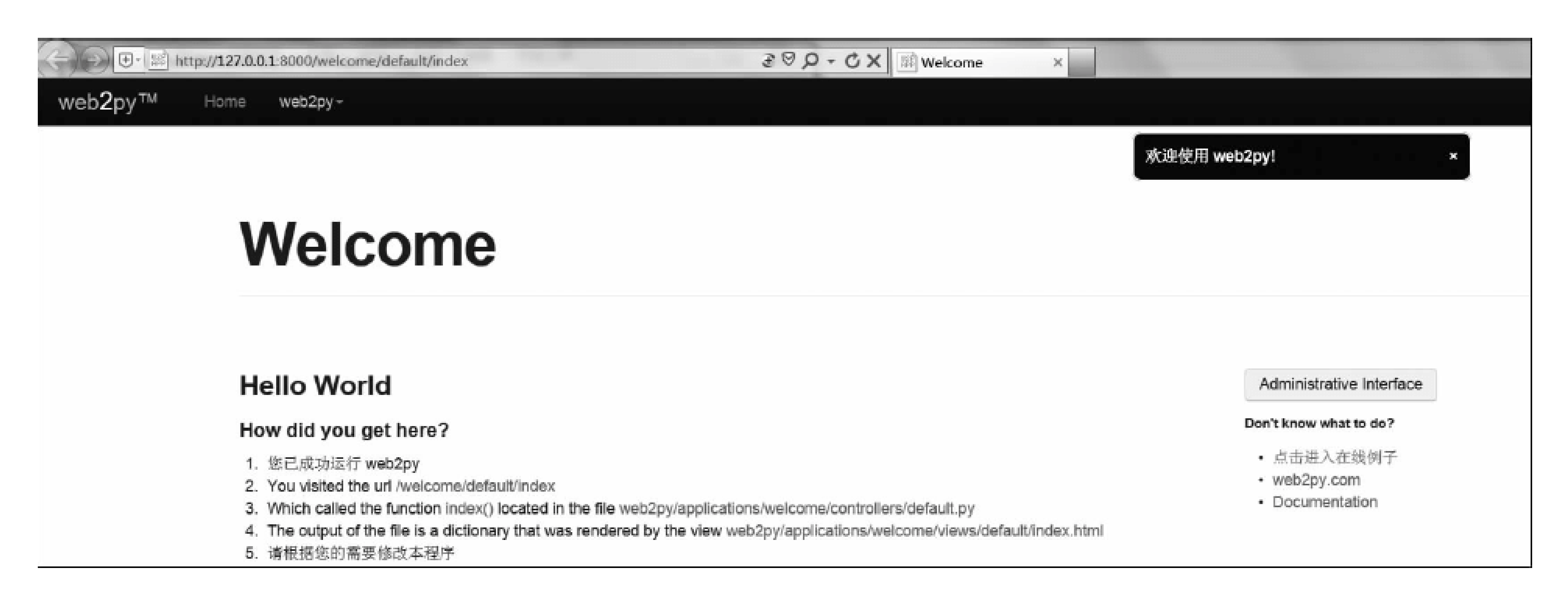


图 17.5　web2py 的欢迎界面

单击图 17.5 右边的 Administrative Interface，输入设置的密码，进入 web2py 的管理员界面，如图 17.6 所示。在 web2py 中，每个网站都是一个应用，也就是一个 App，默认的应用有 admin、examples 和 welcome 3 个。

【例 17-1】　在网页上展示一个输入框，输入姓名 zhou，单击提交按钮，跳转到另一页面，显示“Hello zhou”。

【解析】　具体步骤如下所示：

步骤 1：创建应用 myapp，如图 17.7 所示。

应用 myapp 的开发界面如图 17.8 所示。

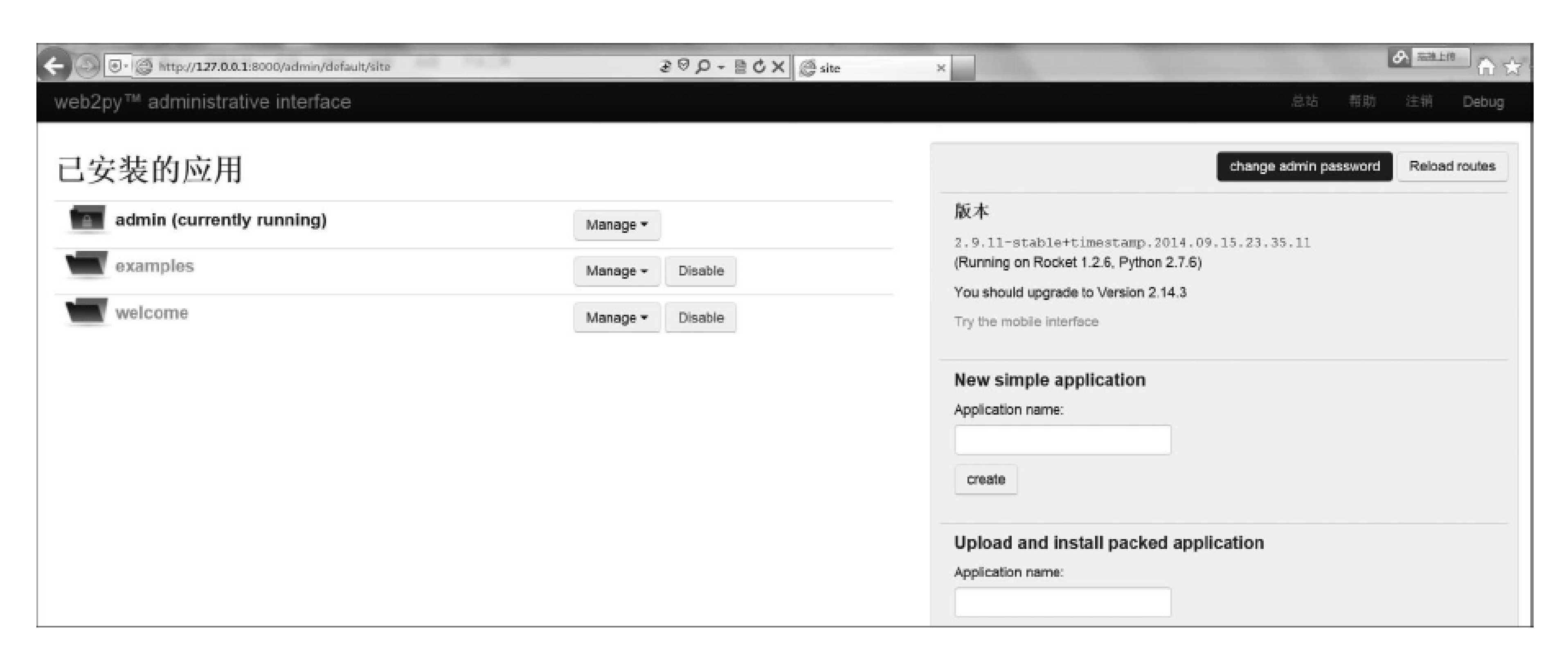


图 17.6 web2py 的管理员界面

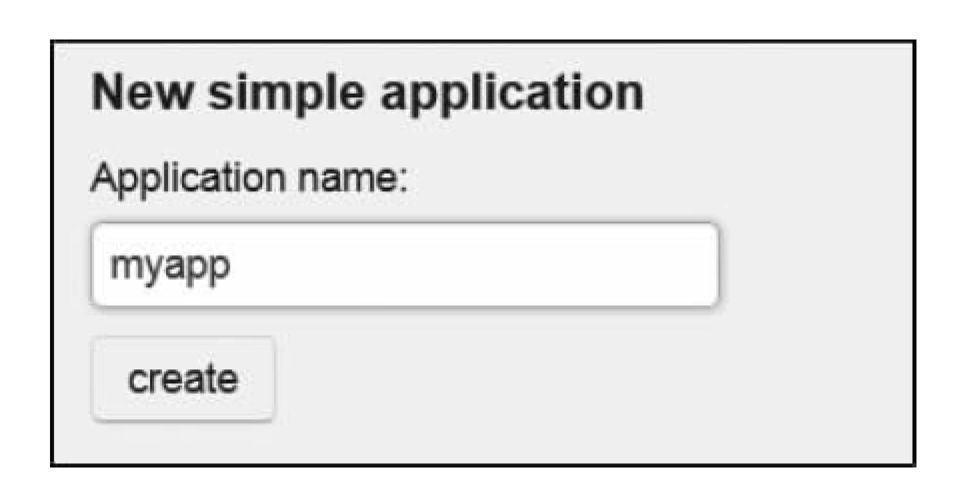


图 17.7 创建应用 myapp

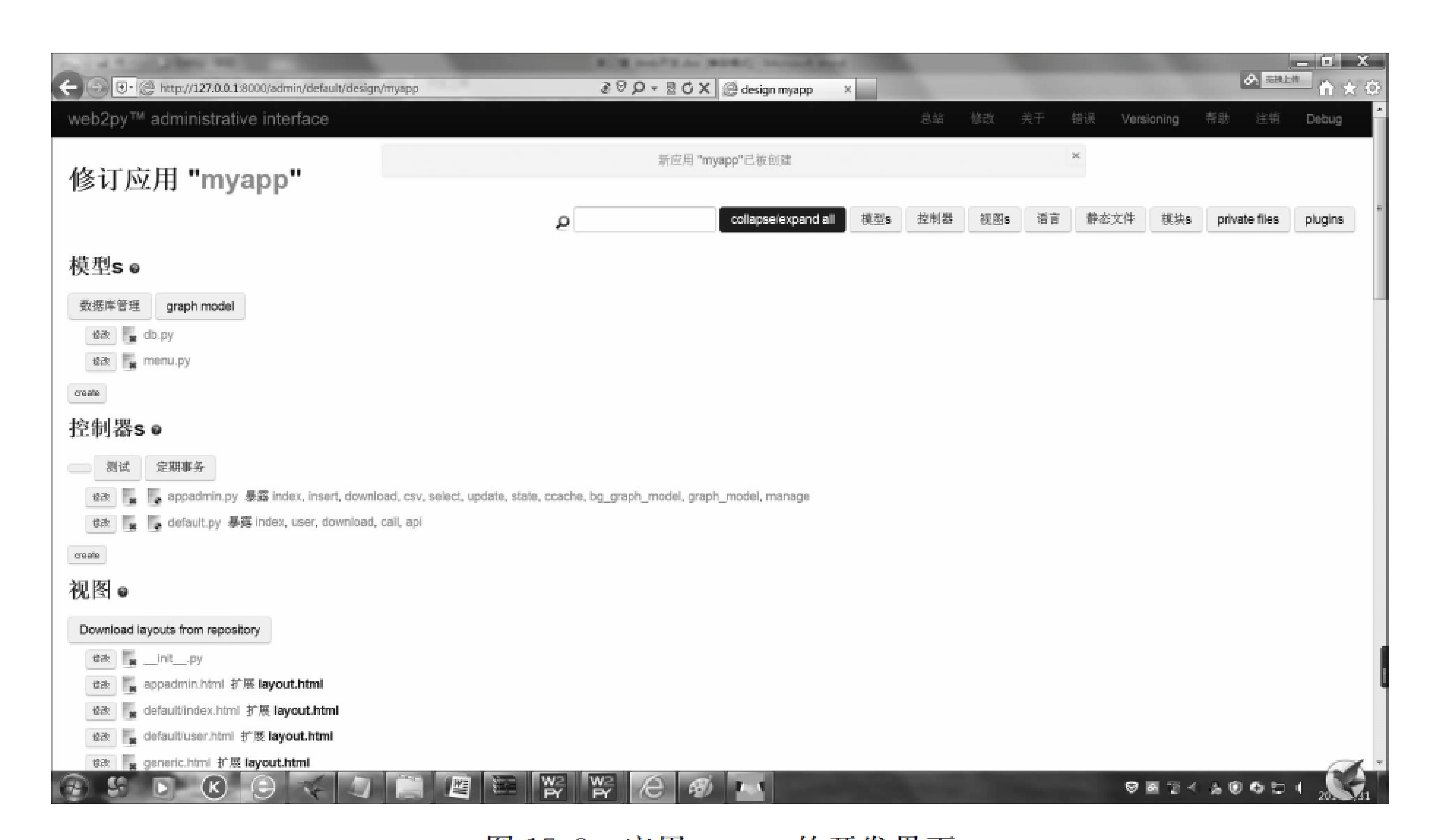


图 17.8 应用 myapp 的开发界面

步骤 2：分别创建 MVC 的模型、控制器和视图。

(1) 创建模型。

模型是程序要处理的数据，开发者只需要将数据库的表定义好，web2py 会自动完成

表的创建、数据的查询、删除和插入等数据库操作。通过修改 model 下的 db. py 文件来创建表，打开 db. py，可以看到里面有许多系统事先生成的代码，不要修改这些代码，在文件的最后追加以下内容，完成 image 表的定义，该表只有一个 titile 字段。

```
# db.py
db.define_table('image',
          Field('title'))
```

在浏览器中输入 http://127. 0. 0. 1:8000/myapp/appadmin/index，就可以看到 web2py 的数据库管理界面，如图 17. 9 所示。auth_user 和 auth_group 等以“auth_”开头的表都是系统表，在最后可以看到创建的表 image，单击其旁的“新记录”按钮，出现如图 17. 10 所示的添加新记录界面，可以向表 image 中添加数据。



图 17. 9　web2py 的数据库管理界面

由于本实例功能较为简单，不需要任何代码，因此只需要设计“控制器”和“视图”的相关代码。

(2) 创建控制器。

单击 create 按钮，输入 sayhello，如图 17. 11 所示。

在 sayhello. py 文件中输入如下代码：

```
# - * - coding: utf- 8- * -
#尝试
```

Myapp Database Administration (appadmin)

数据库 db 数据表 image

新记录

Title:

提交

图 17.10　添加新记录界面



图 17.11　创建控制器

```
def index(): return dict(message="hello from sayhello.py")
def first():
    form=FORM(INPUT(_name='visitor_name',requires=IS_NOT_EMPTY()),
    INPUT(_type='submit'))
    if form.process().accepted:
        session.visitor_name=form.vars.visitor_name
        redirect(URL('second'))
    return dict(form=form)

def second():
    name=session.visitor_name or redirect(URL('first'))
    return dict(name=name)
```

代码说明：

- first()函数中包括输入框，命名为 visitor_name 和 submit 型的按钮。
- second()函数中的 name 是从 session. visitor_name 中读取的。

函数的输出可以是字符串或符号字典(哈希表)。如果用户请求 HTML 页面(默认情况),字典将被呈现为 HTML 页面。如果用户以 XML 请求同一页面,web2py 将会尝试找到一个能将字典呈现为 XML 格式的视图。

(3) 创建视图。

单击 create 按钮,输入 sayhello/first. html 和 sayhello/second. html 两个文件,如图 17.12 所示。



图 17.12　创建视图

在 sayhello/first. html 和 sayhello/second. html 两个文件中分别输入如下代码:

```
# sayhello/first.html
{{extend 'layout.html'}}
<h1>请输入您的姓名</h1>
{{= form}}

# sayhello/second.html
{{extend 'layout.html'}}
<h1>Hello{{= session.visitor_name or "anonymous"}}</h1>
```

视图的代码是由 HTML 和 Python 混合写成的,“{{…}}”中为 Python 代码。

步骤 3:运行程序。

在浏览器中输入 http://127. 0. 0. 1:8000/myapp/sayhello/first,程序运行界面如图 17.13 所示。

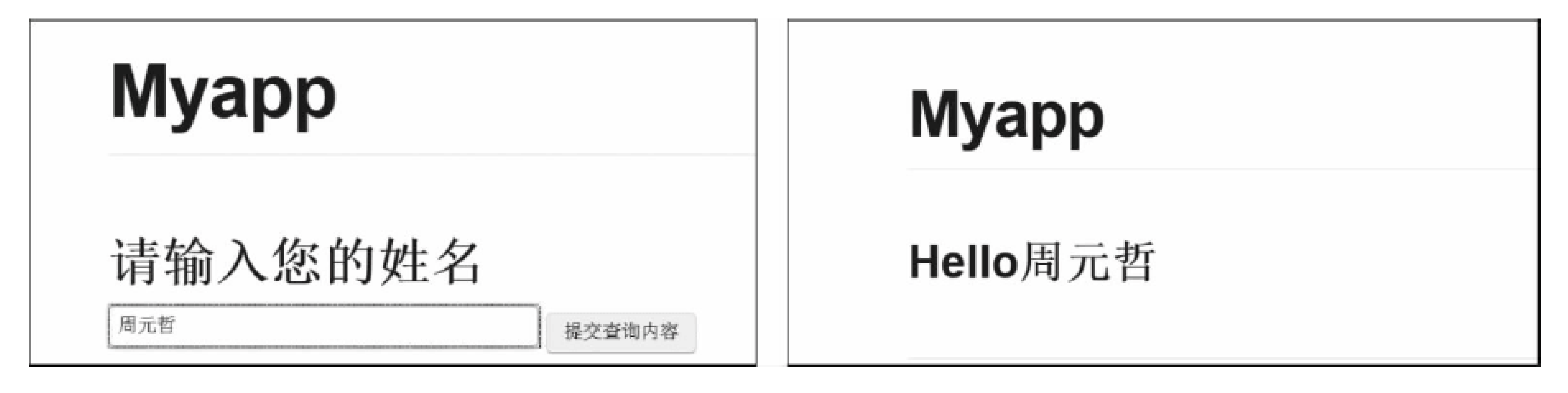


图 17.13　程序运行界面

17.2.3　Django 框架

Django 是用 Python 语言编写的开源的 Web 应用框架。

1. 安装 Django

安装 Django 的步骤如下:

步骤 1:下载 Django 并安装。

在网址 https://www.djangoproject.com/download/中下载 Django，版本为 Django-1.2.7，将其解压缩到与 Python 相同的根目录下，在命令行窗口中进入 Django-1.2.7 目录，执行如下安装命令：

```
python setup.py install
```

执行过程如图 17.14 所示。

```
c:\>cd C:\Django-1.2.7

C:\Django-1.2.7>python  setup.py install
```

图 17.14　安装 Django

步骤 2：配置系统环境变量。

Django 将要被安装到 Python 的 lib/site-packages 目录下，配置环境变量如下：

```
C:/Python27/Lib/site-packages/django;C:/Python27/Scripts
```

执行过程如图 17.15 所示。



图 17.15　配置系统环境变量

步骤 3：检查是否安装成功。

(1) 在命令行窗口中输入 python。

(2) 输入 import django。

(3) 输入 django.get_version()。

执行过程如图 17.16 所示。

```
c:\>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.get_version()
'1.2.7'
>>>
```

图 17.16　检测安装

2. 在 PyCharm 中新建 Django 工程

创建 myFirstDjango 工程，如图 17.17 所示。

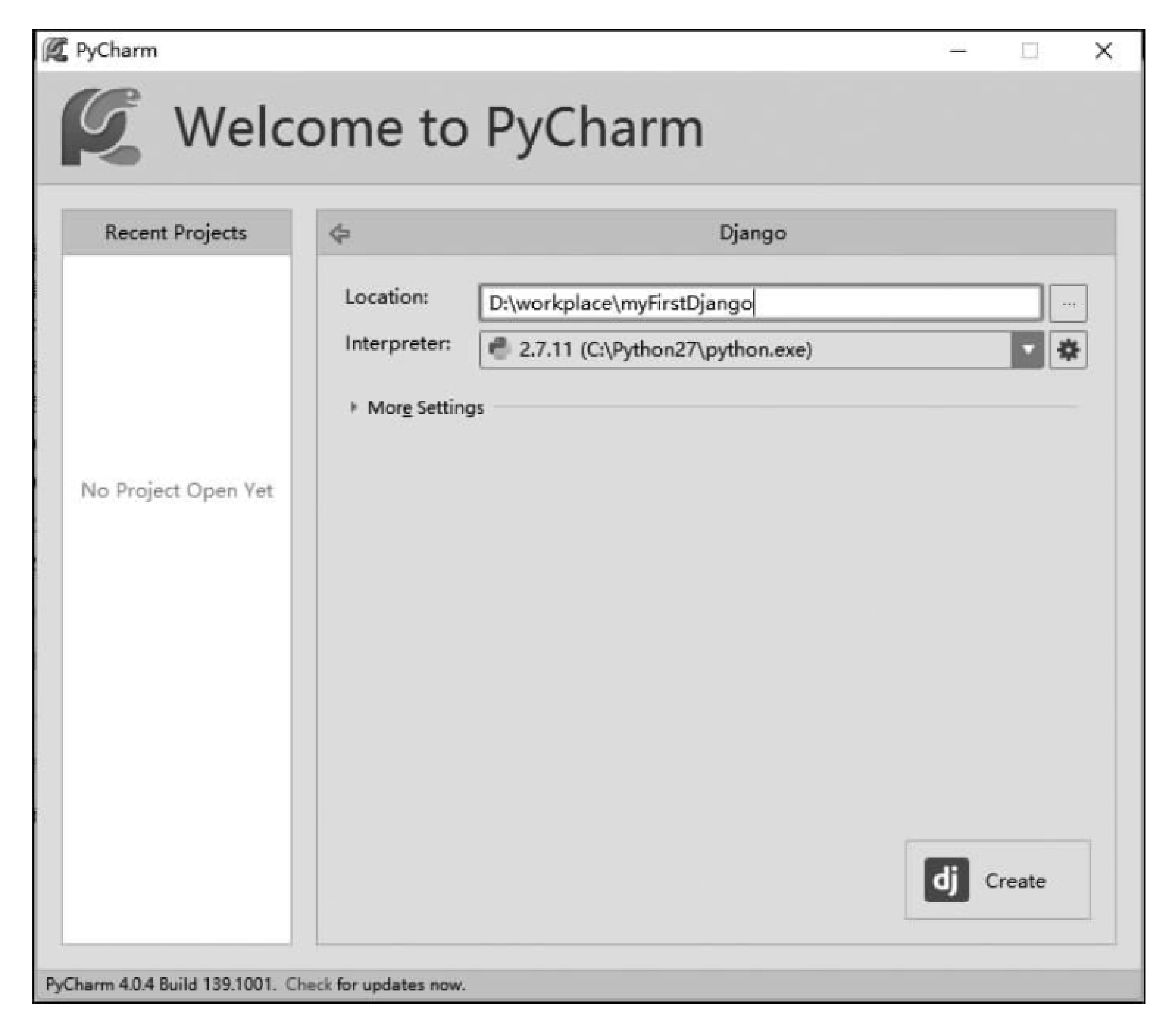


图 17.17　创建 myFirstDjango 项目截图 1

单击 create 按钮，出现如图 17.18 所示。

myFirstDjango 项目的文件框架如下：

myFirstDjango 下表示工程的全局配置，分别为__init__. pysetttings. py、urls. py 和 wsgi. py。其中：

- settings. py 为该 Django 项目的数据库配置、应用配置和其他配置。
- urls. py 为该 Django 项目的 Web 工程 URL 映射的配置。
- __init__. py 是让 Python 把 mysite 目录当作一个包。
- wsgi. py 是 Web 服务器的入口。

templates 目录为模板文件的目录。

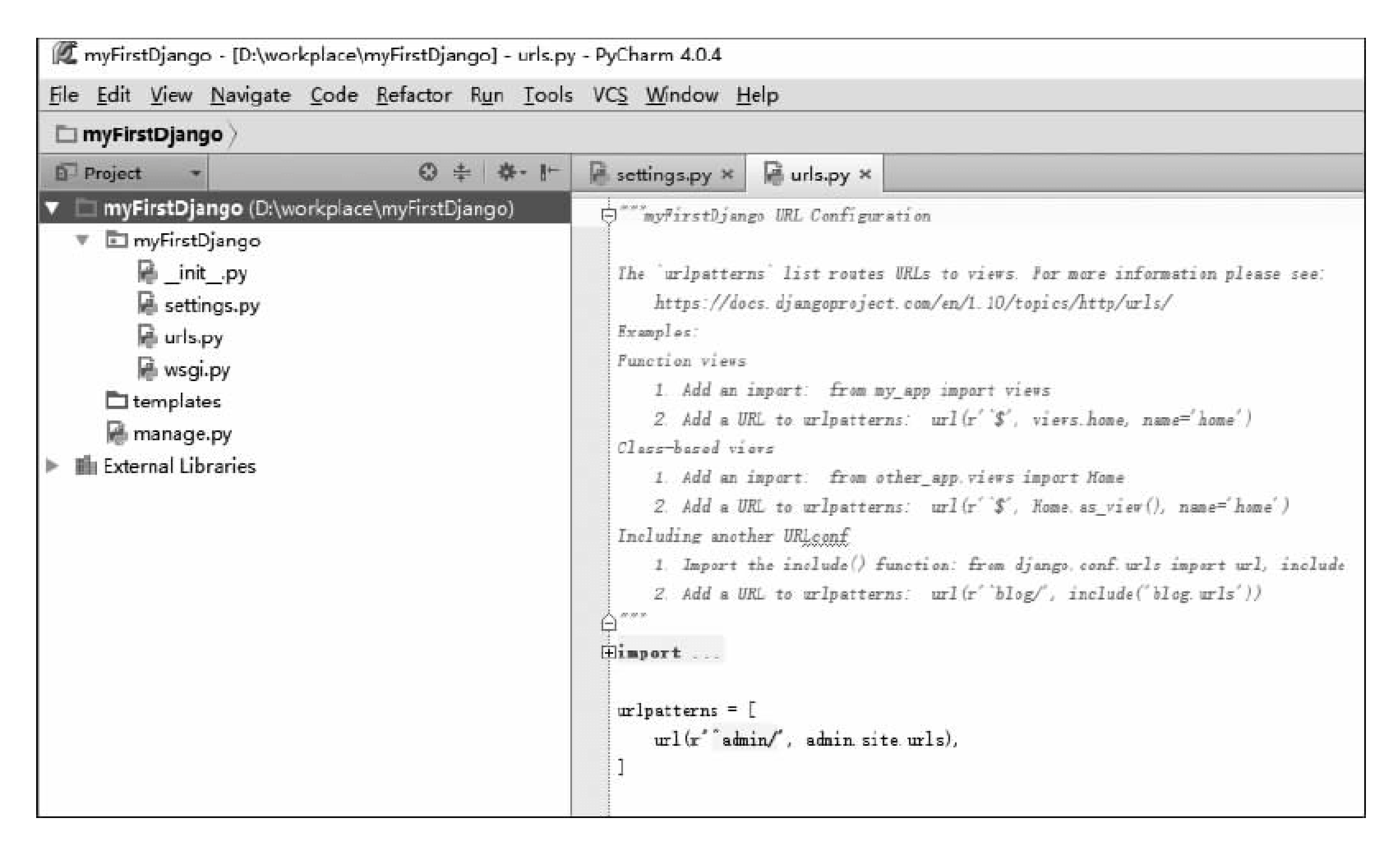


图 17.18 创建 myFirstDjango 项目截图 2

manage.py 是 Django 提供的一个管理工具，可以同步数据库等。

创建完成后，单击 Run 按钮，如图 17.19 所示。

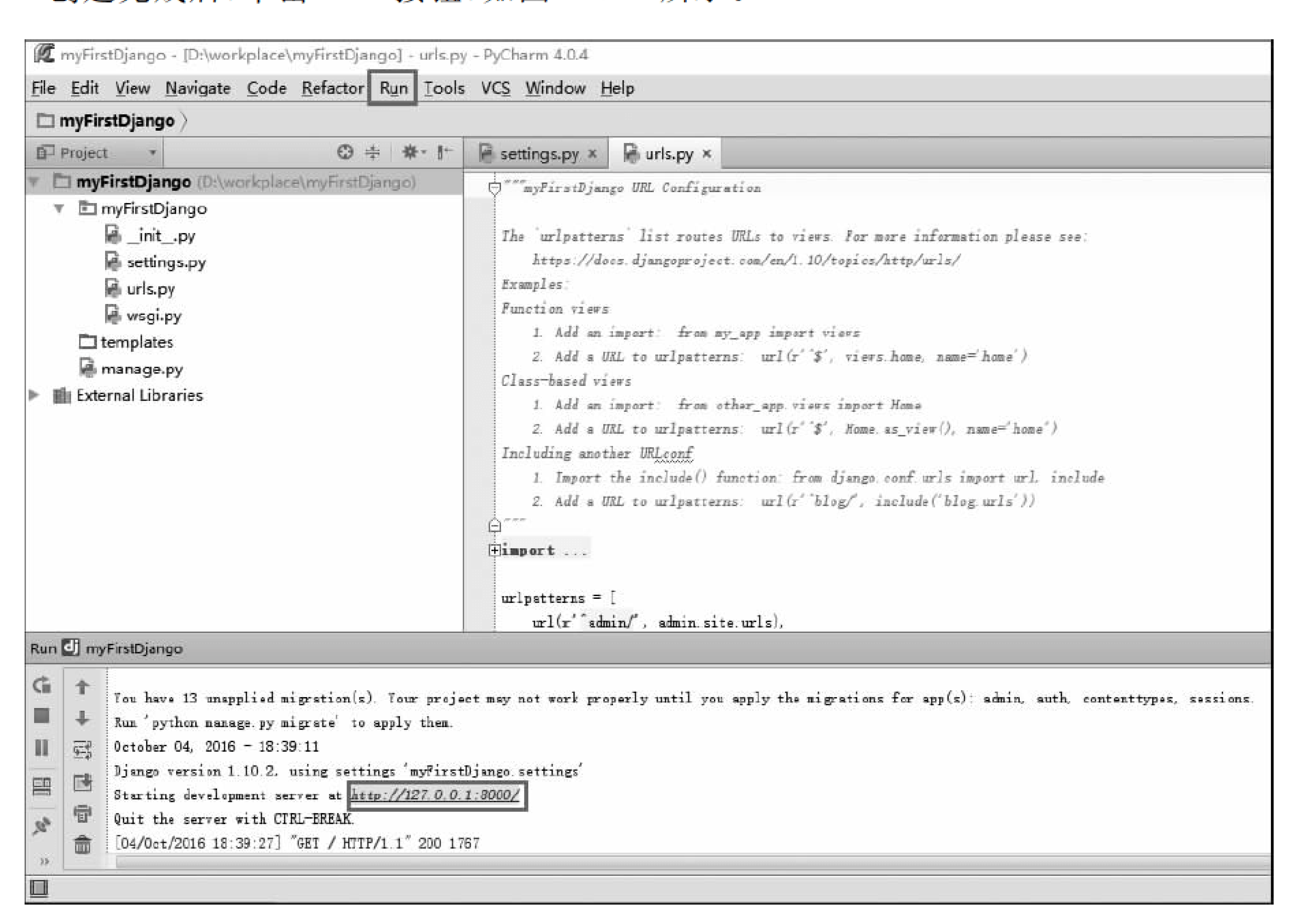


图 17.19 运行 myFirstDjango 项目

单击图 17.19 中 http://127.0.0.1:8000/链接，出现如图 17.20 所示的页面。

【例 17-2】 实现在网页上输出 Hello World 的功能。

首先，新建 views.py 文件，内容如图 17.21 所示。

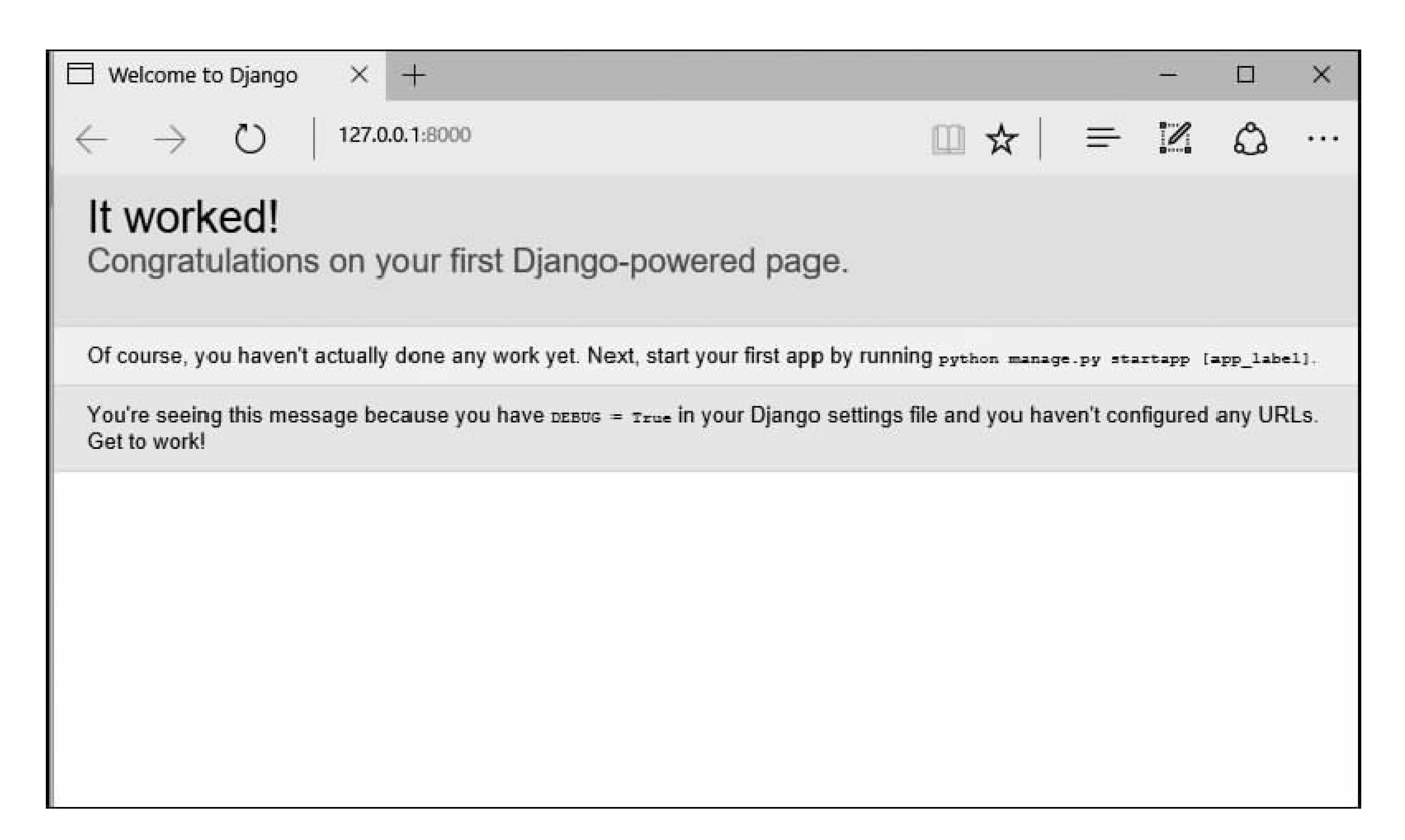


图 17.20 在浏览器上输入 127.0.0.1:8000

```
settings.py × urls.py × views.py ×
__author__ = 'Pow'
from datetime import *
import datetime
from django.http import HttpResponse
def hello(request):
    s = 'Hello World!'
    current_time = datetime.datetime.now()
    html = '<html><head></head><body><h1> %s </h1><p> %s </p></body></html>' % (s, current_time)
    return HttpResponse(html)
```

图 17.21 views.py 文件

其次，修改 urls.py 文件的映射配置内容，如图 17.22 所示。

```
settings.py × urls.py × views.py ×
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url, include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import url
from django.contrib import admin
from myFirstDjango.views import hello
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^hello/$', hello),
]
```

图 17.22 修改 urls.py 文件

当用户输入 http://127.0.0.1:8000/hello 时，便会调用 hello 方法，该方法通过

HttpResponse()将页面内容作为响应返回,如图 17.23 所示。

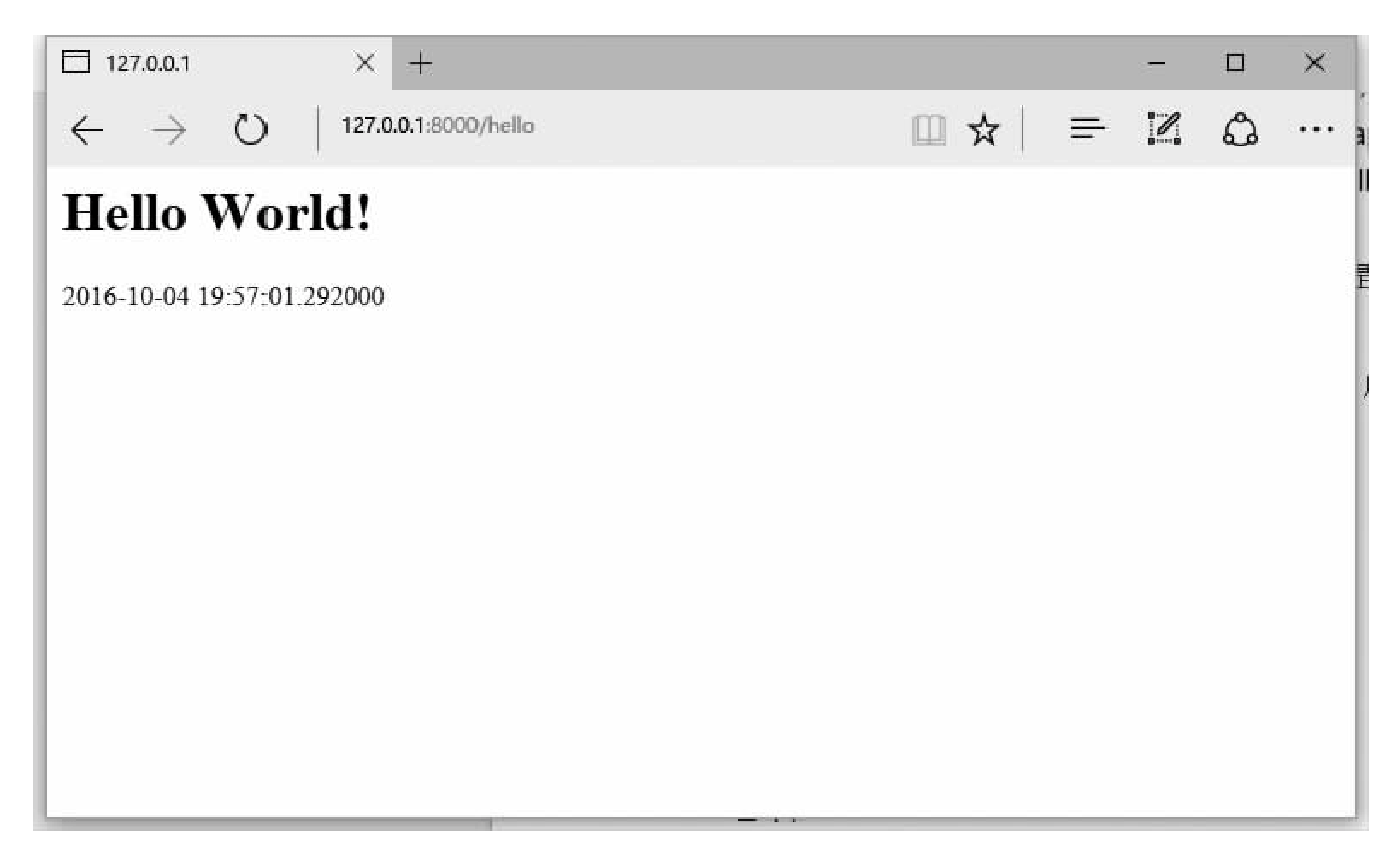


图 17.23 运行 hello

【例 17-3】 显示动态的数据。

例 17-2 中,views.py 调用 HttpResponse()类作为响应返回到浏览器。这样使得页面逻辑和页面表现混合在一起,不但工作量比较大,而且若要展示一些动态的数据,则无法完成。

本例采用如下方法:首先在 templates 目录下新建 index.html 文件,其内容如图 17.24 所示。

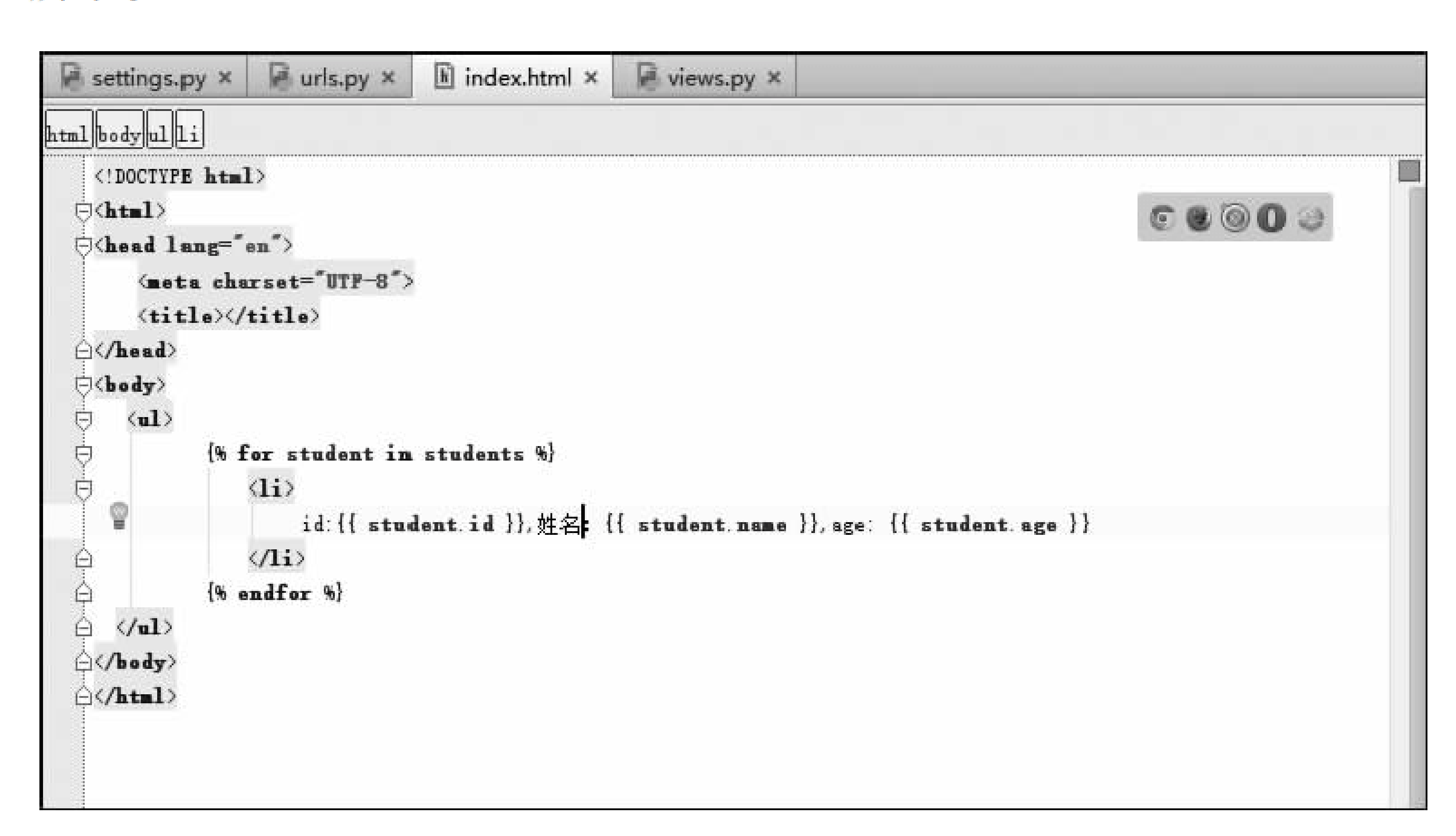


图 17.24 index.html 文件

修改 views.py 文件,添加 show()方法,如图 17.25 所示。

添加映射,修改 urls.py 内容,如图 17.26 所示。

修改 settings.py 模板配置:'DIRS':[BASE_DIR+r'\templates'],如图 17.27 所示。

settings.py × urls.py × index.html × views.py ×

```
__author__ = 'Pow'
from datetime import *
import datetime
from django.http import HttpResponse
def show(request):
    list = [{id: 1, 'name': 'Jack'}, {id: 2, 'name': 'Rose'}]
    return render_to_response('index.html',{'students': list})
```

图 17.25　在 views.py 文件中添加 show()方法

settings.py × urls.py × index.html × views.py ×

```
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url, include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import url
from django.contrib import admin
from myFirstDjango.views import hello
from myFirstDjango.views import show
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^hello/$', hello),
    url(r'^show/$', show),
]
```

图 17.26　修改 urls.py 内容

settings.py × urls.py × index.html × views.py ×

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR+r'\templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

图 17.27　修改 settings.py

运行后会出错，如图 17.28 所示。

出错后，解决办法是在 views.py 添加 from django.shortcuts import render_to_response，如图 17.29 所示。

最后运行结果如图 17.30 所示。

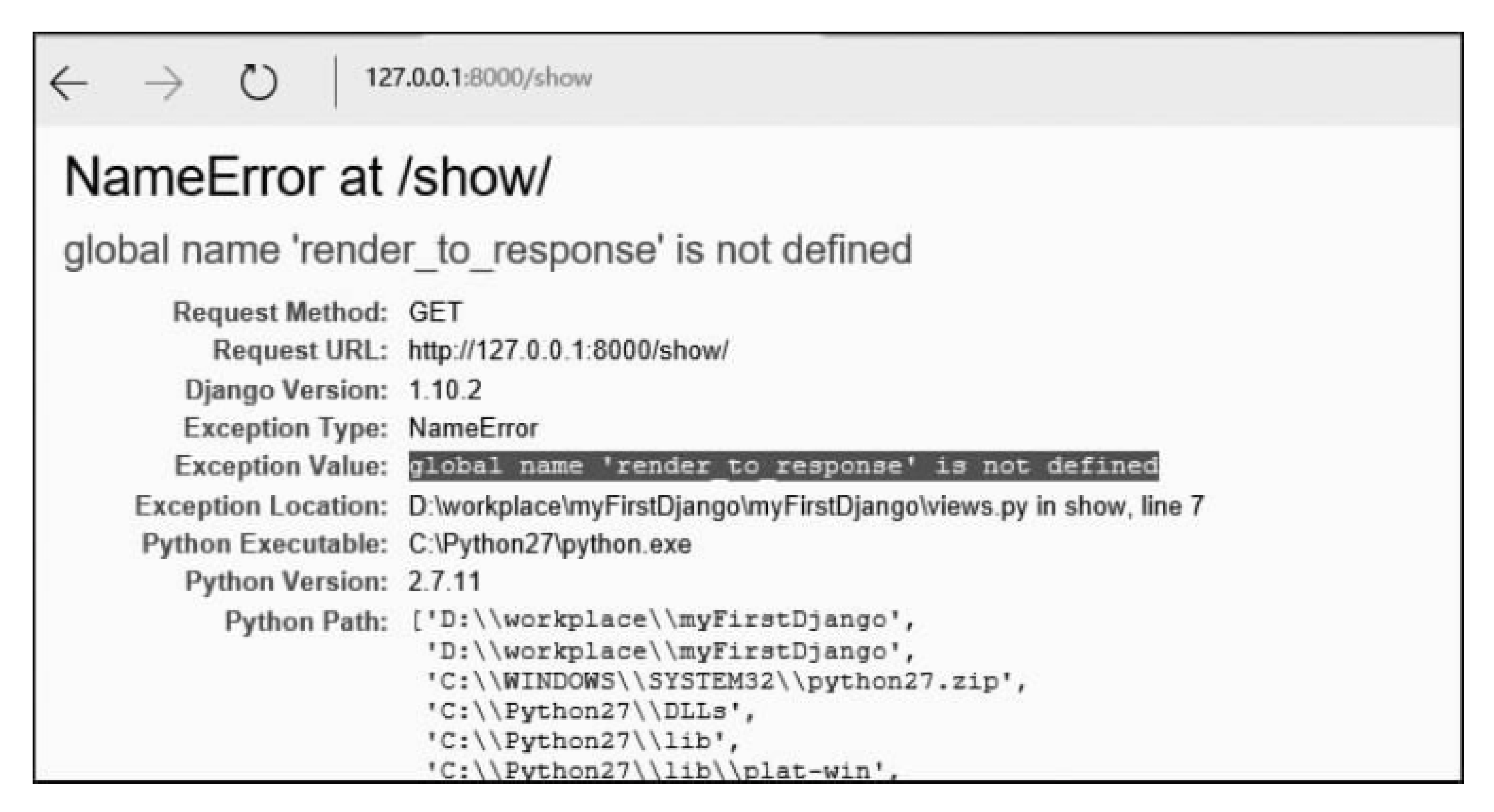


图 17.28　运行错误

settings.py × | urls.py × | index.html × | views.py ×

```
__author__ = 'Pow'
from datetime import *
import datetime
from django.http import HttpResponse
from django.shortcuts import render_to_response
def show(request):
    list = [{id: 1, 'name': 'Jack'}, {id: 2, 'name': 'Rose'}]
    return render_to_response('index.html',{'students': list})
```

图 17.29　修改 views.py 文件

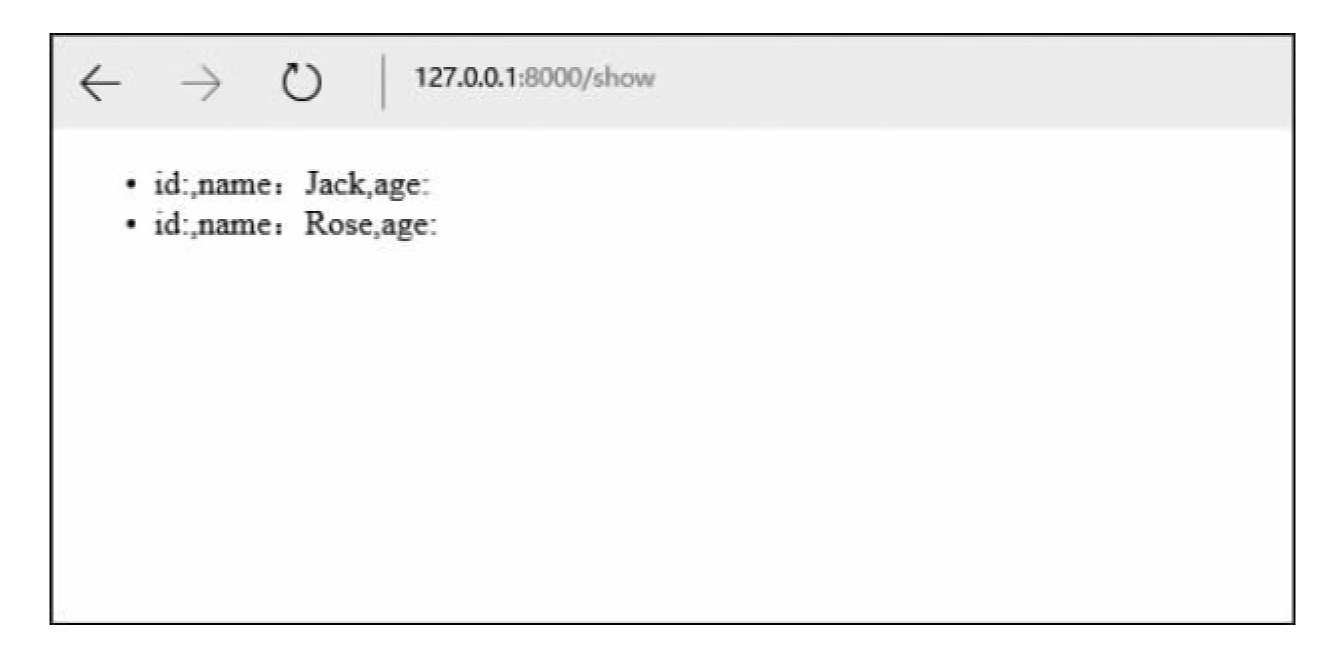
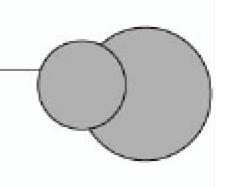


图 17.30　运行结果

第18章 游戏开发



18.1 本章要求

- 掌握 pygame 的下载和安装。
- 掌握 pygame 的相关模块。

18.2 本章知识重点

18.2.1 游戏简介

游戏是互动娱乐性的一类软件,用户可以通过各种输入输出设备,通过设计的特定流程或规则进行交互和游戏体验。最早的游戏主要运行在特定的游戏机上,在 PC 逐渐普及后开始转向 PC 平台,进而向网页和移动设备发展。随着游戏的发展,玩家对游戏的体验不仅局限于玩法,对画面的要求也不断提升,于是便出现了游戏引擎的概念。游戏引擎集成了计算机图形学和音频动画等模块,用于对图片和特效进行渲染,从而达到出色的表现效果。

游戏分为 2D 游戏和 3D 游戏。2D 游戏主要集中于策略类和横版过关类,视角固定,注重玩法。而 3D 游戏出现以后,让玩家更生动地接触到开发者设计的环境中,游戏体验更为自由。随着游戏进一步的发展,诸如 UnrealEngine(虚幻)、CryEngine、FrostbiteEngine(寒霜)、NaughtyDogEngine(顽皮狗)、LunimousStudio(夜光)等用于开发大型 3D 游戏的引擎被开发出来,集成了大量的碰撞特效、粒子效果、光源、物理特效等,开发出诸如《质量效应》《战地》《孤岛危机》《神秘海域》等游戏,画面高度逼近现实,特效场面宏大,具有精美的表现力。近来随着交互设备的技术性革新,从光学体感设备到 VR、AR 的出现,势必会掀起游戏开发的新一轮机遇和发展。

18.2.2 pygame 简介

Python 的 pygame 模块专门用于电子游戏设计,其下载网址为 http://www.pygame.org/download.shtml,如图 18.1 所示。

本书 Python 的版本为 2.7.3,选择 pygame-1.9.1.win32-py2.7.msi 下载,安装后输入如下命令检验是否安装成功。

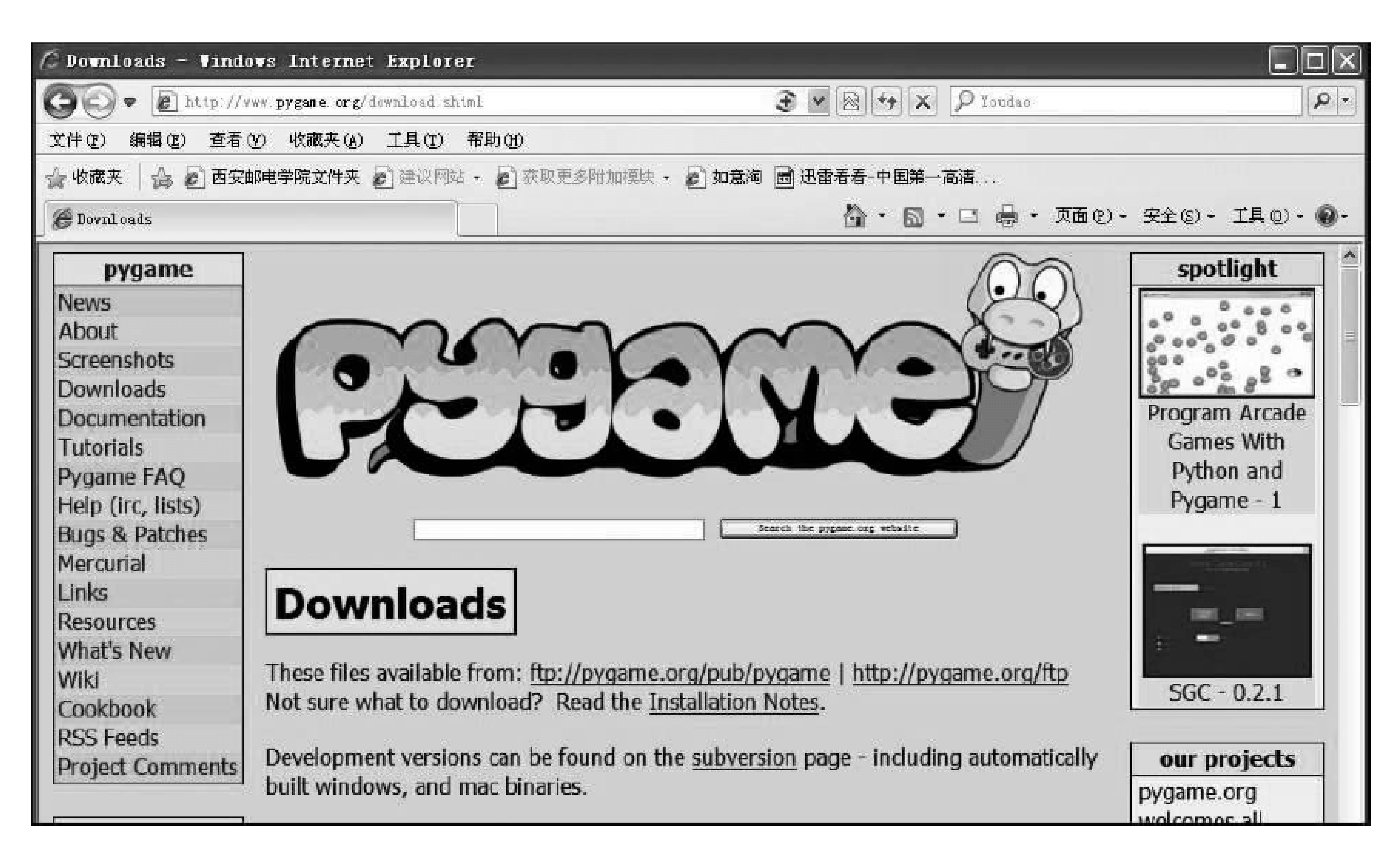


图 18.1　Pygame 下载网址

```
>>> import pygame
>>> print pygame.ver
```

如果安装成功,则会显示 pygame 的版本号,如图 18.2 所示。

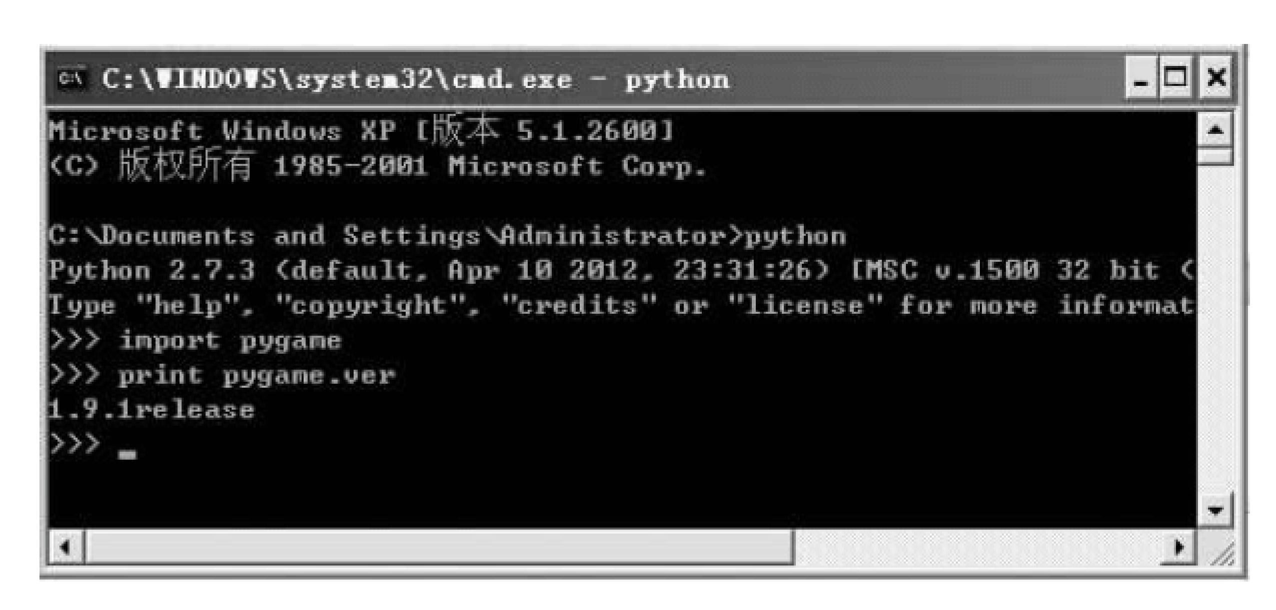


图 18.2　测试 pygame 安装是否成功

pygame 一般通过 while 循环反复检测是否有事件产生,根据事件来更新游戏状态,具体完成如下操作:

- 处理事件。
- 更新游戏状态。
- 在屏幕上绘图。

【例 18-1】 Pygame 版本的 HelloWorld 程序。

```
import pygame, sys                                  #调用 pygame 模块和 sys 模块
from pygame.locals import *
pygame.init()                                       #初始化 pygame
DISPLAYSURF=pygame.display.set_mode((400, 300))     #建立宽 400 像素、高 300 像素的窗口
pygame.display.set_caption('Hello World!')          #设置标题栏上的文本
```

```
    while True:                                  #无限循环
        for event in pygame.event.get():         #获取各种键盘及鼠标事件
            if event.type==QUIT:
                pygame.quit()
                sys.exit()
        pygame.display.update()
```

程序运行结果如图 18.3 所示，仅在窗口的标题上显示了“Hello World!”，当单击窗口的关闭按钮，程序将会退出。



图 18.3 程序运行结果

18.2.3 pygame 模块

pygame 中有很多模块，每个模块对应不同的功能，如表 18.1 所示。

表 18.1 pygame 模块

模块名	功能	模块名	功能
pygame. cdrom	访问光驱	pygame. movie	播放视频
pygame. cursors	加载光标	pygame. music	播放音频
pygame. display	访问显示设备	pygame. overlay	访问高级视频叠加
pygame. draw	绘制形状、线和点	pygame. rect	管理矩形区域
pygame. event	管理事件	pygame. sndarray	操作声音数据
pygame. font	使用字体	pygame. sprite	操作移动图像
pygame. image	加载和存储图片	pygame. surface	管理图像和屏幕
pygame. joystick	使用游戏手柄或者类似的设备	pygame. surfarray	管理点阵图像数据
pygame. key	读取键盘按键	pygame. time	管理时间和帧信息
pygame. mixer	声音	pygame. transform	缩放和移动图像
pygame. mouse	鼠标		

下面详细介绍鼠标、键盘、绘图和动画。

1. 鼠标

【例 18-2】 鼠标举例。

```
import os, pygame
from pygame.locals import *
from sys import exit
from random import *
if not pygame.font:print('Warning, Can not found font!')
pygame.init()                                        #模块初始化
screen=pygame.display.set_mode((255, 255), 0, 32)
screen.fill((255,255,255))
font=pygame.font.SysFont('stcaiyun',20)
text=font.render('Cliked Me please!!!', True, (34, 252, 43))
mouse_x, mouse_y=0, 0
while 1:
    for event in pygame.event.get():
        if event.type==QUIT:
            exit()
        elif event.type==MOUSEBUTTONDOWN:
            pressed_array=pygame.mouse.get_pressed()
            for index in range(len(pressed_array)):
                if pressed_array[index]:
                    if index==0:
                        print('Pressed LEFT Button!')
                    elif index==1:
                        print('The mouse wheel Pressed!')
                    elif index==2:
                        print('Pressed RIGHT Button!')
        elif event.type==MOUSEMOTION:
            #return the X and Y position of the mouse cursor
            pos=pygame.mouse.get_pos()
            mouse_x=pos[0]
            mouse_y=pos[1]
    screen.fill((mouse_x, mouse_y, 0))
    screen.blit(text, (40, 100))
    pygame.display.update()
```

程序运行结果如图 18.4 所示，当鼠标经过窗口的时候，窗口背景颜色会随着鼠标的移动而发生改变，当鼠标单击窗口，在控制台打印出是鼠标的哪个键被单击了：左，右，滚轮。

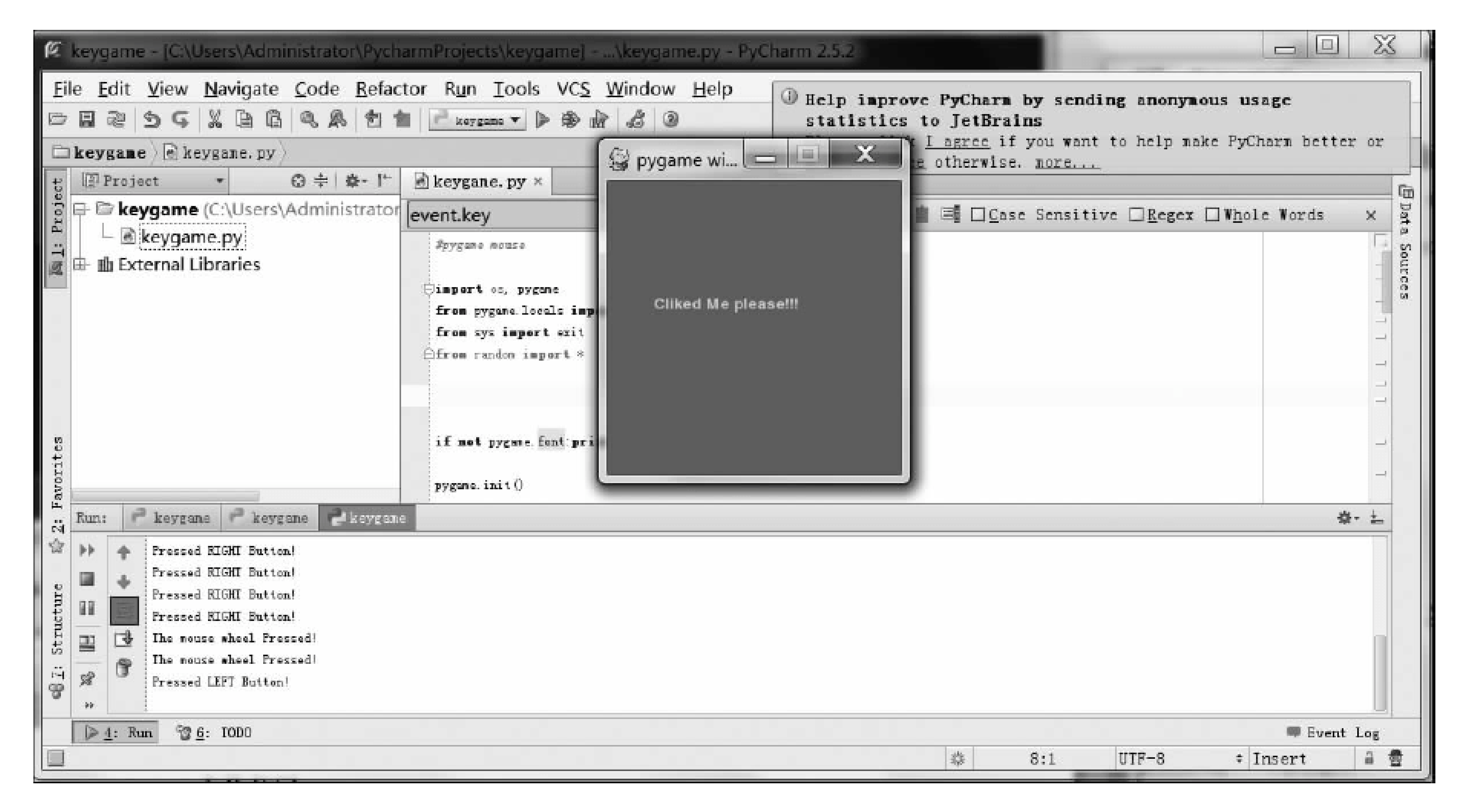


图 18.4　例 18-2 程序运行结果

2. 键盘

【例 18-3】 键盘举例。

```
import pygame
from pygame.locals import *
from sys import exit
BG_IMAGE= 'C:\\Python27\\plane.jpg'                  #飞机图片
pygame.init()
screen= pygame.display.set_mode((1000, 2000), 0, 32)   #屏幕大小设置
bg= pygame.image.load(BG_IMAGE).convert()
x, y= 0, 0
move_x, move_y= 0, 0
while 1:
    for event in pygame.event.get():
        #print(event.type)
        if event.type= =QUIT:
            exit()
        if event.type= =KEYDOWN:
            print(event.key)
            if event.key= =K_LEFT:                     #左方向键按下的处理
                move_x= -100
            elif event.key= =K_UP:                     #上方向键按下的处理
                move_y= -100
            elif event.key= =K_RIGHT:                  #右方向键按下的处理
                move_x= 100
            elif event.key= =K_DOWN:                   #下方向键按下的处理
```

```
            move_y=100
    elif event.type==KEYUP:
        move_x=0
        move_y=0
x+=move_x
y+=move_y
#print(x, y)
screen.fill((0, 0, 0))
screen.blit(bg, (x, y))
pygame.display.update()
```

程序运行结果如图 18.5 所示，飞机图片会随着上、下、左、右方向键而移动位置。

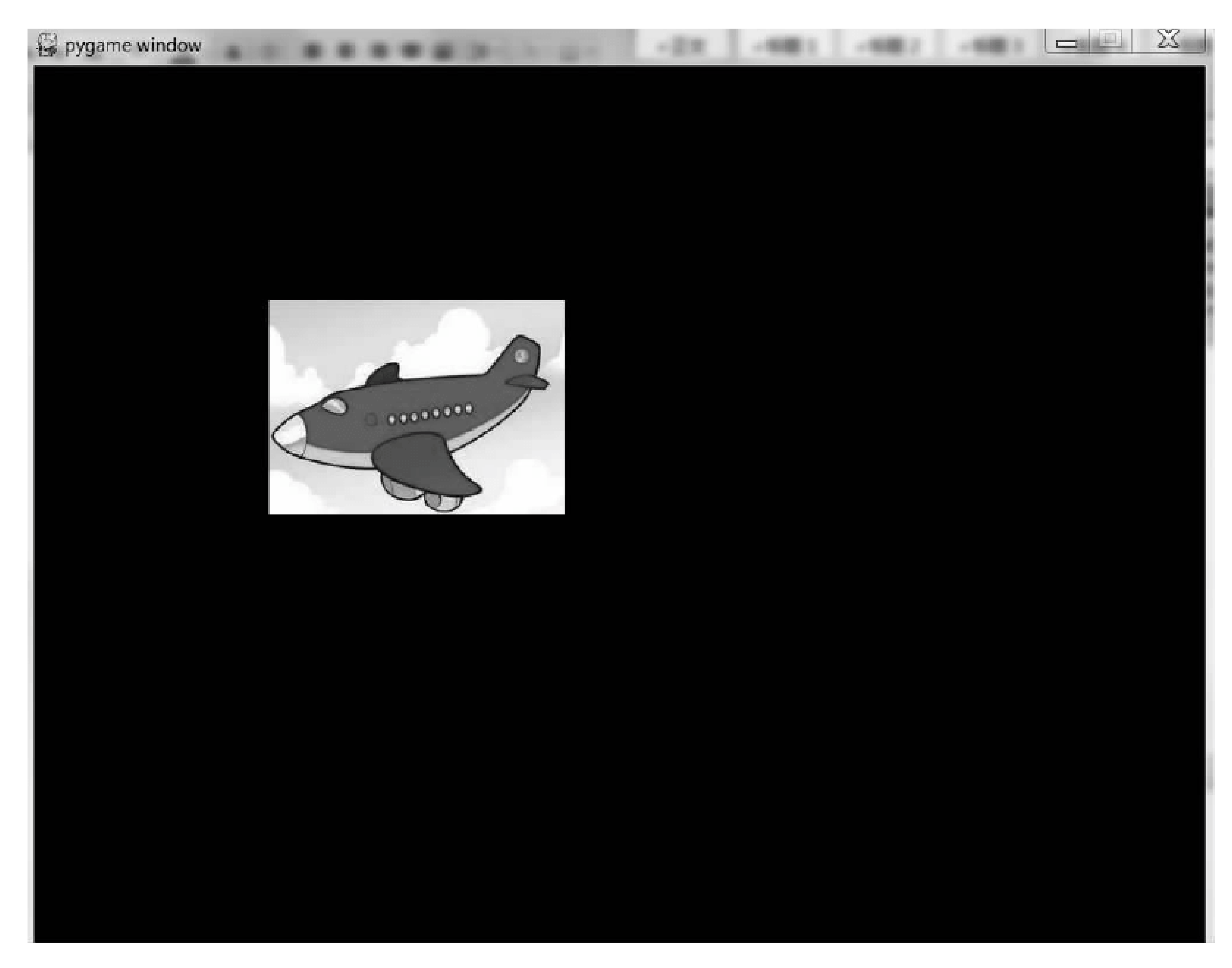


图 18.5　例 18-3 程序运行结果

3. 绘图

【例 18-4】 绘图举例。

```
import pygame, sys
from pygame.locals import *
pygame.init()
windowSurface=pygame.display.set_mode((500, 400), 0, 32)
pygame.display.set_caption("hello, world")
BLACK=(0, 0, 0)
WHITE=(255, 255, 255)
```

```
RED=(255, 0, 0)
GREEN=(0, 255, 0)
BLUE=(0, 0, 255)
basicFont=pygame.font.SysFont(None, 48)
text=basicFont.render("Hello,world", True, WHITE, BLUE)
textRect=text.get_rect()
textRect.centerx=windowSurface.get_rect().centerx
textRect.centery=windowSurface.get_rect().centery
windowSurface.fill(BLACK)
pygame.draw.polygon(windowSurface, GREEN, ((146, 0),(291, 106), (236, 277), (56, 277), (0, 106)))
pygame.draw.line(windowSurface, BLUE, (60, 60), (120,60), 4)
pygame.draw.line(windowSurface, BLUE, (120, 60), (60,120))
pygame.draw.line(windowSurface, BLUE, (60, 120), (120,120), 4)
pygame.draw.circle(windowSurface, BLUE, (300, 50), 20, 0)
pygame.draw.ellipse(windowSurface, RED, (300, 250, 40,80), 1)
pygame.draw.rect(windowSurface, RED, (textRect.left-20, textRect.top-20, textRect.width+40, textRect.
height+40))
pixArray=pygame.PixelArray(windowSurface)
pixArray[480][380]=BLACK
del pixArray
windowSurface.blit(text, textRect)
pygame.display.update()
while True:
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
            sys.exit()
```

程序运行结果如图 18.6 所示。

4. 动画

帧动画即通过一系列静态图辅之以连续快速变化产生动画效果。依据这一原理制作动画,一般需要考虑如下几个因素:

（1）时间,什么时间移动,多长时间变下一个动作。

（2）位置,从什么位置到什么位置。

（3）动作,前后两个动作的连续性。

【例 18-5】 弹球示例。

```
import sys
import pygame
from pygame.locals import *
def play_ball():
    pygame.init()
    #窗口大小
```

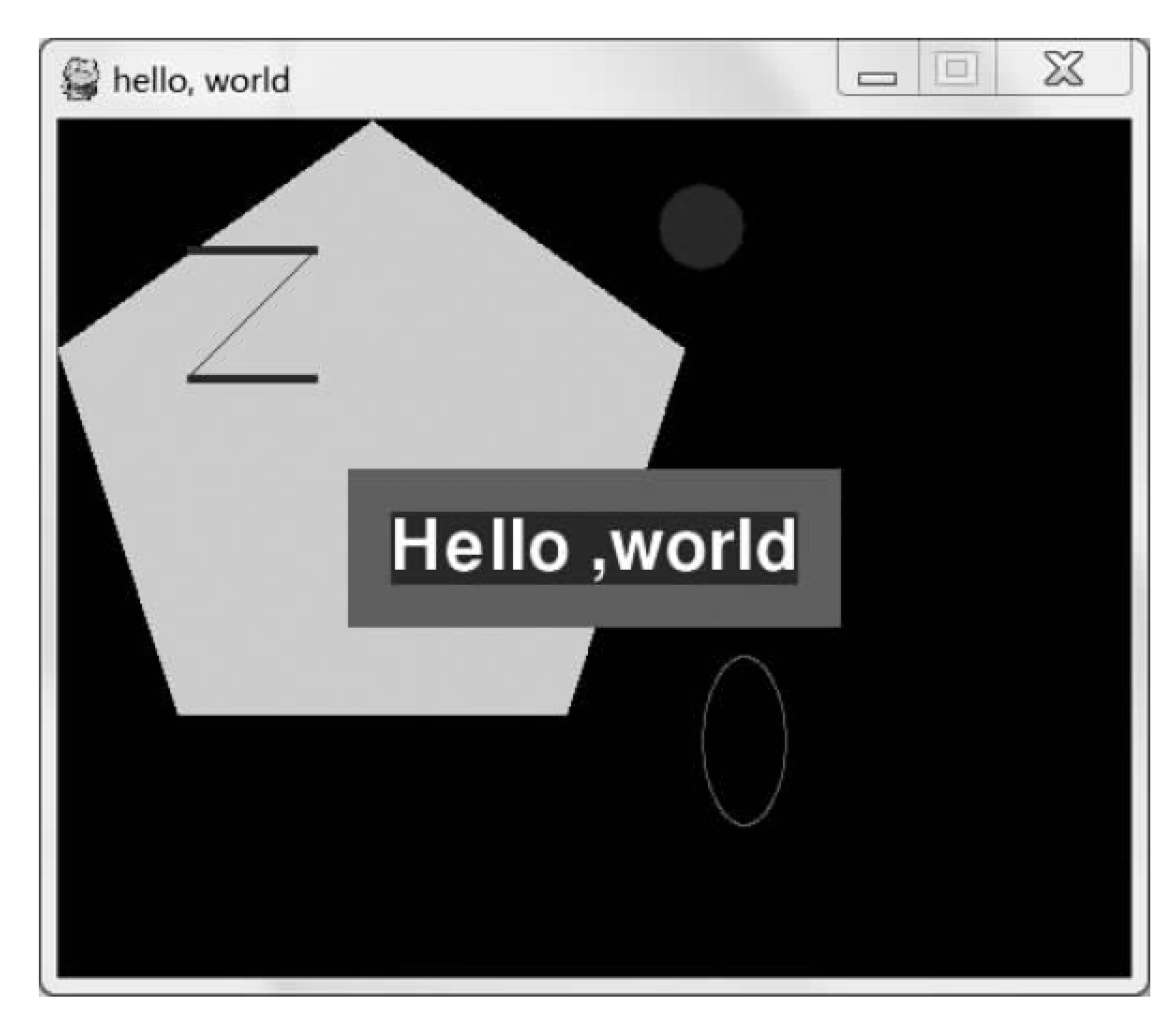


图 18.6 例 18-4 程序运行结果

```
window_size= (width, height)= (1000,800)
#小球运行偏移量[水平,垂直],值越大,移动越快
speed= [1, 1]
#窗口背景色 RGB值
color_black= (0, 0, 139)
#设置窗口模式
screen=pygame.display.set_mode(window_size)
#设置窗口标题
pygame.display.set_caption('运动的小球')
#加载小球图片
ball_image=pygame.image.load('C:\\Python27\\ball.jpg')
#获取小球图片的区域形状
ball_rect=ball_image.get_rect()
while True:
#退出事件处理
    for event in pygame.event.get():
        if event.type= =pygame.QUIT:
            pygame.quit()
            sys.exit()
            #使小球移动,速度由 speed变量控制
    ball_rect=ball_rect.move(speed)
    #当小球运动出窗口时,重新设置偏移量
    if (ball_rect.left < 0) or (ball_rect.right > width):
        speed[0]= - speed[0]
    if (ball_rect.top < 0) or (ball_rect.bottom > height):
        speed[1]= - speed[1]
```

```
        #填充窗口背景
        screen.fill(color_black)
        #在背景 Surface 上绘制小球
        screen.blit(ball_image, ball_rect)
        #更新窗口内容
        pygame.display.update()
if __name__=='__main__':
    play_ball()
```

程序运行截图如图 18.7 所示，篮球的图片会自动在蓝色屏幕持续进行碰撞运动。

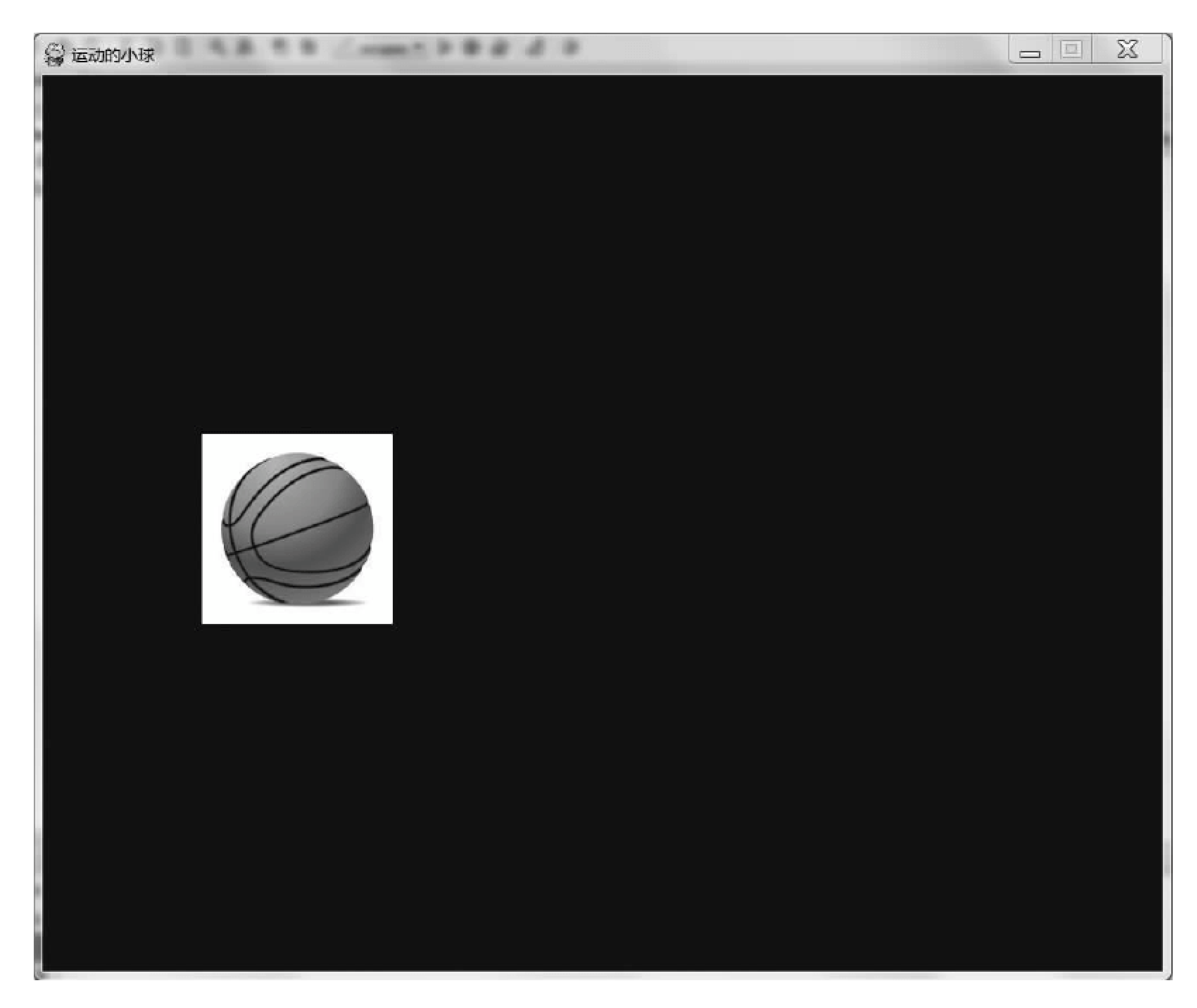


图 18.7 例 18-5 程序运行截图

参 考 文 献

[1] 周元哲．Python 程序设计基础．北京：清华大学出版社，2015．

[2] 李文新，郭炜，余华山．程序设计导引及在线实践．北京：清华大学出版社，2007．

[3] 周元哲．Visual Basic. NET 程序设计．西安：西安电子科技大学出版社，2014．

[4] 周元哲．Visual Basic 程序设计语言．北京：清华大学出版社，2011．

[5] Swaroop C H．简明 Python 教程．沈洁元，译．shttp://old. sebug. net/paper/python/．

[6] 周元哲，刘伟，邓万宇．程序基本算法教程．北京：清华大学出版社，2016．

[7] 李文新，郭炜，余华山．程序设计导引及在线实践．北京：清华大学出版社，2007．

[8] 裘宗燕．从问题到程序：程序设计与 C 语言引论．北京：机械工业出版社，2011．

[9] 赵家刚，狄光智，等．计算机编程导论：Python 程序设计．北京：人民邮电出版社，2013．

[10] 沙行勉．计算机科学导论——以 Python 为舟．北京：清华大学出版社，2014．

[11] William F. Punch，Richard Enbody．Python 入门经典．张敏，等译．北京：机械工业出版社，2012．

[12] Python 官方网站．http://www. python. org/．

[13] Django 官方网站．https://www. djangoproject. com/．